1993年2月1日発行(毎月1回1日発行)第12巻2号通巻130号 昭和58年11月2日第三種郵便物認可

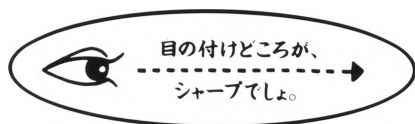
PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

特集 画像創造のために

1992年度GAME OF THE YEARノミネート発表
モデリングコンバータCAD_CNV.BAS/アクセラレータ(その2)
新製品紹介 Communication SX-68K/版下作成Y300-A

2
1993

オー!エックス
定価600円

SHARP

"感性"咲かせるワー

POWER WORKSTATION

インテリジェントなパフォーマンスを誇るX68000Compact XVIと多彩にラインアップされたペリフェラル。感性を刺激するクリエイティブなワークステーション環境が自在に構築できます。

- パーソナルワークステーション(2HD3.5インチFDDタイプ・本体＋キーボード＋マウス)
  CZ-674C-H(グレー) 標準価格**298,000**円(税別)
- 15型カラーディスプレイテレビ
  CZ-614D-TN(チタンブラック)・**-BK**(ブラック) 標準価格**135,000**円(税別)
  ■ディスプレイテレビ/CZ-6TU用RGBケーブル**CZ-6CR1** 標準価格**4,500**円(税別)
  ■ディスプレイテレビ/CZ-6TU用TVコントロールケーブル**CZ-6CT1** 標準価格**5,500**円(税別)
- 80MB内蔵用ハードディスクドライブ
  CZ-68HA 好評発売中
- 5.25インチ増設用フロッピーディスクドライブ
  CZ-6FD5 標準価格**99,800**円(税別・接続ケーブル同梱)
- 光磁気ディスクユニット
  CZ-6MO1 標準価格**450,000**円(税別)
  ■SCSI変換ケーブル**CZ-6CS1** 標準価格**12,000**円(税別)
- 2MB増設RAMボード
  CZ-6BE2D 標準価格**54,800**円(税別・取り付け費別)
  ■2MB増設RAM**CZ-6BE2B** 標準価格**54,800**円(税別・取り付け費別)×2
  ■数値演算プロセッサ**CZ-6BP2** 標準価格**45,800**円(税別・取り付け費別)
- 48ドット熱転写カラー漢字プリンタ
  CZ-8PC5-BK(ブラック) 標準価格**96,800**円(税別)
- MIDIボード
  CZ-6BM1A 標準価格**26,800**円(税別)
- インテリジェントコントローラ
  CZ-8NJ2 標準価格**23,800**円(税別)

拡張スロットへ
MIDI楽器へ
本体内へ

68買ったらEXEクラブへ入ろう!

EXEクラブって何だ?

X68000を手に入れたら、やっぱり他のユーザーがどんな風に使っているのか気になるもの。ということでEXEクラブは、そんなあなたのための、他の68ユーザーとのコミュニケーションをバックアップする、情報交換の場です。

資料請求券
X68000ペリフェラル
Oh!X
2月

ァステーション環境。

GRAPHIC WORKSTATION

- パーソナルワークステーション(2HD3.5インチFDDタイプ・本体＋キーボード＋マウス)
  CZ-674C-H(グレー) 標準価格**298,000**円(税別)
- 21型カラーディスプレイ **CU-21HD** 標準価格**148,000**円(税別)
- 80MB内蔵用ハードディスクドライブ **CZ-68HA** 好評発売中
- 光磁気ディスクユニット **CZ-6MO1** 標準価格**450,000**円(税別)
  ■SCSI変換ケーブル**CZ-6CS1** 標準価格**12,000**円(税別)
- 2MB増設RAMボード **CZ-6BE2D** 標準価格**54,800**円(税別・取り付け費別)
  ■2MB増設RAM**CZ-6BE2B** 標準価格 **54,800**円(税別・取り付け費別)×2
  ■数値演算プロセッサ**CZ-6BP2** 標準価格**45,800**円(税別・取り付け費別)
- カラーイメージスキャナ
  CZ-8NS1 標準価格**188,000**円(税別)
  ■スキャナ用パラレルボード**CZ-6BN1** 標準価格**29,800**円(税別)
- カラーイメージジェット
  IO-735X-B(ブラック) 標準価格**248,000**円(税別)
  ■接続ケーブル**IO-73CX** 標準価格**5,500**円(税別)

STANDARD WORKSTATION

- パーソナルワークステーション
  (2HD3.5インチFDDタイプ・本体＋キーボード＋マウス)**CZ-674C-H**(グレー) 標準価格**298,000**円(税別)
- 14型カラーディスプレイ**CZ-608D-H**(グレー) 標準価格**94,800**円(税別)
- 5.25インチ増設用フロッピーディスクドライブ**CZ-6FD5** 標準価格**99,800**円(税別・接続ケーブル同梱)

TFT COLOR LCD WORKSTATION

- パーソナルワークステーション
  (2HD3.5インチFDDタイプ・本体＋キーボード＋マウス)**CZ-674C-H**(グレー) 標準価格**298,000**円(税別)
- 10.4型カラー液晶ディスプレイ**LC-10C1-H**(グレー) 標準価格**598,000**円(税別)
  ■接続ケーブル**AN-1515X** 標準価格**4,200**円(税別)

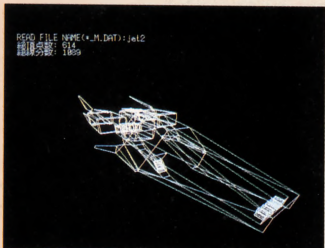
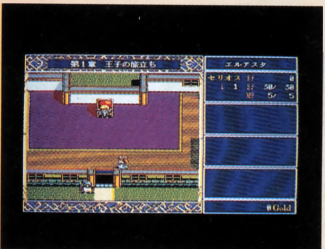
※カラー液晶ディスプレイを接続してご使用の場合、SX-WINDOW上のアプリケーション利用に限定されます。

本体同梱の入会申込ハガキを送るだけで、自動的に無料入会。さらに下記の特典付き。

メリット1 会員ナンバー入りオリジナル会員電卓がもらえる。

メリット2 各種フェアご優待・イベント案内等、数々の特典がある。

●お問い合わせは…
シャープ株式会社
電子機器事業本部システム機器営業部
〒545 大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)
電子機器事業本部AVCシステム事業推進室
〒162 東京都新宿区市谷八幡町８番地☎(03)3260-1161(大代表)

特集　画像創造のために

モデリングコンバータCAD_CNV.BAS

ドラゴンスレイヤー英雄伝説

キングス・ダンジョン

DōGA CGアニメーション講座

（で）のショートプロぱーてぃ

Oh!X

C O N T

〈スタッフ〉
●編集長／前田　徹　●副編集長／植木章夫　●編集／浅井研二　山田純二　豊浦史子　●協力／有田隆也　中森　章　林　一樹　吉田幸一　華門真人　吉田賢司　影山裕昭　大和　哲　村田敏幸　丹　明彦　三沢和彦　長沢淳博　宮島　靖　金子俊一　浦川博之　石上達也　柴田　淳　御木徳高　瀧　康史　●カメラ／杉山和美　●イラスト／山田晴久　寺尾響子　高橋哲史　川原由唯　●アートディレクター／島村勝頼　●レイアウト／元木昌子　ADGREEN　●校正／グループごじら

表紙絵：須藤　牧人

E　N　T　S

1993 FEB. 2

■広告目次

SHARP

X68000 CompactXVI NEWS

Opinion 1

（ハードディスクが使いたい。）

Compact専用の内蔵ハードディスクが登場しました。SCSI仕様の80MB。場所を取らずに高速・大容量ファイル環境を実現します。

■内蔵用ハードディスクドライブ（CZ-674C専用）
CZ-68HA……………………………好評発売中
※取りつけに関してはシャープお客様ご相談窓口にてご相談ください（取りつけ費別）。

さらに大容量をお望みの場合、外付け用のSCSI端子で一般のSCSIハードディスクも接続可能。フルピッチSCSI端子とハーフピッチSCSI端子を接続するためのSCSI変換ケーブルも用意しています。

■SCSI変換ケーブル
CZ-6CS1………標準価格12,000円（税別）

CZ-6CS1

Opinion 2

（従来のソフト資産を活かしたい。）

これについても、Compact専用の外付け5インチフロッピーディスクユニットを用意していますから、従来の68シリーズの資産を有効活用できます。3.5インチと5インチの間でのデータのやりとりも可能。また、CZ-674C及びCZ-6FD5のスイッチ設定を変えれば、5インチソフトからの起動が可能になり、市販ソフトなどそのまま使えます。

■増設用5インチ・フロッピーディスク・ユニット（CZ-674C専用）
CZ-6FD5……………………標準価格99,800円（税別）

Opinion 3

（ディスプレイテレビを接続したい。）

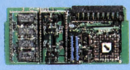
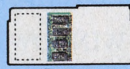
Compactは、従来のシリーズと比べ体積比44%と小さいため、コネクタの形状も異なっていますが、このケーブルを使用することにより、ディスプレイテレビやRGBシステムチューナーを利用できます。

CZ-6CR1
CZ-6CT1

■15型カラーディスプレイテレビ（スピーカー・チルトスタンド同梱）
CZ-614D-TN……………標準価格135,000円（税別）

■ディスプレイテレビ/CZ-6TU用RGBケーブル
CZ-6CR1…………………標準価格　4,500円（税別）

■ディスプレイテレビ/CZ-6TU用テレビコントロールケーブル
CZ-6CT1…………………標準価格　5,500円（税別）

資料請求券
X68000ペリフェラル
Oh!X
2係

パーソナルワークステーション X68000 Compact XVIについてのご意見、ご要望にお応えします。

Opinion 4

（メモリ環境をパワーアップしたい。）

Compactは2MBのメインメモリを標準装備していますが、本体内で最大8MBまで拡張できます。

	容量	周辺機器
標準	2MB	――
拡張	4MB	CZ-6BE2D
	6MB	CZ-6BE2B
	8MB	CZ-6BE2B×2

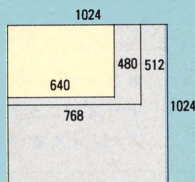
■2MB増設RAMボード　CZ-6BE2D　標準価格54,800円（税別）

■2MB増設RAM　CZ-6BE2B　標準価格54,800円（税別）

※取りつけに関してはシャープお客様ご相談窓口にてご相談ください（取りつけ費別）。

Opinion 5

（液晶ディスプレイとSX-WINDOWの関係は？）

液晶ディスプレイ（LC-10C1-H 標準価格598,000円・税別）の解像度は640×480ドット。Compactでは、従来のX68000シリーズの画面モードにこの画面モードをプラス。解像度の制約を受けないウィンドウ環境ならではの機能です。このようにSX-WINDOW環境の確立により、ハードウェアに依存しない快適な操作環境が実現します。

Opinion 6

（数値演算プロセッサはほんとに速い？）

ご存じのようにMPU68000自体は複雑な計算（浮動小数点演算）を単純な計算の組み合わせで行っています。X68000シリーズに装備されている浮動小数点演算パッケージ「FLOAT2.X」は、よく使う単純な組み合わせをまとめたもの。数値演算プロセッサは、いわばこのパッケージの機能を、ハードウェアで高速に実現し、MPUの負担を軽くするものです。アプリケーションプログラムの中には浮動小数点演算を必要としないものもあるため、すべてのプログラムが高速になるわけではありませんが、レイトレーシングなど大量の実数演算を必要とするソフトウェアの場合、飛躍的な実行速度の向上が期待できます。

■数値演算プロセッサ CZ-6BP2 標準価格45,800円（税別）

※数値演算プロセッサはCZ-6BE2D上に装着します。
※取りつけに関してはシャープお客様ご相談窓口にてご相談ください（取りつけ費別）。

X68000
PERSONAL WORKSTATION・XVI
Compact

本体＋キーボード＋マウス
2HD3.5インチFDDタイプ　CZ-674C-H（グレー）　標準価格298,000円（税別）
14型カラーディスプレイ（ドットピッチ0.28mm）
CZ-608D-H（グレー）　標準価格94,800円（税別）

■お問い合わせは…電子機器事業本部システム機器営業部　〒545　大阪市阿倍野区長池町22番22号　☎(06) 621-1221（大代表）
電子機器事業本部AVCシステム事業推進室　〒162　東京都新宿区市谷八幡町8番地　☎(03)3260-1161（大代表）
シャープ株式会社

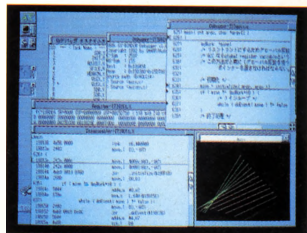
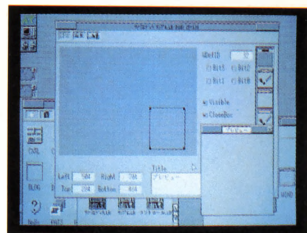
目の付けどころが、
シャープでしょ。

待望のSX-WINDOW開発支援ツール、登場。

SX-WINDOW 開発キット Workroom SX-68K

CZ-288LWD 開発中

SX-WINDOW用のソフト開発に必要な開発ツールやサンプルプログラムを装備。プログラムの編集、リソースの作成、コンパイル、デバッグといった一連の作業をSX-WINDOW上で効率よく実行できます。初めてSX-WINDOW用のプログラムに挑戦する人にも、簡単に基本機能の理解ができる33種のサンプルプログラム付き。また各マネージャ解説と関数リファレンスの詳細なマニュアルも装備しています。

※本ソフトのご使用に際しては、メインメモリ4MB以上、SX-WINDOW ver2.0以上、C compiler PRO-68K ver 2.1が必要です。

キット構成

■開発ツール

●SXデバッガ
SX-WINDOW上で複数のプログラムを同時にデバッグすることができるソースコードデバッガ。

●リソースエディタ
SX-WINDOW上のリソースをリソースタイプごとの編集ウィンドウでビジュアルに作成・編集が可能。

●リソースリンカ
Cコンパイラやアセンブラで作成したリソースデータファイル(オブジェクトファイル)をリンクしてリソースファイルを作成。

●サンプルメイク
サンプルプログラムのコンパイル作業をSX-WINDOW上から、XCver2.1のMAKE.Xを呼び出して、自動実行する簡易メイクユーティリティ。

■サンプルプログラム

●基礎編(23種)
各マネージャの基本的な機能のみを用いた基本動作の理解。

●応用編(4種)
基礎編での基本機能を応用した簡単なアプリケーションの作成。

●実用編(6種)
基礎/応用編での機能を駆使した、実用的なアプリケーションの作成。

■その他のファイル

●インクルードファイル
Cコンパイラとアセンブラ用の関数定義、データ定義ファイル。

●ライブラリファイル
Cコンパイラ用の関数ライブラリ。

マニュアル

●ユーザーズマニュアル●プログラマーズマニュアル●ファンクションリファレンス●ライブラリリファレンス

X68000
PERSONAL WORKSTATION・XVI
Compact

本体+キーボード+マウス
2HD3.5インチFDDタイプ
CZ-674C-H(グレー) 標準価格298,000円(税別)

14型カラーディスプレイ(ドットピッチ0.28mm)
CZ-608D-H(グレー) 標準価格94,800円(税別)

●5.25インチ増設用フロッピーディスクドライブ
CZ-6FD5
標準価格99,800円・税別
〔接続ケーブル同梱〕

●ディスプレイテレビ/CZ-6TU用RGBケーブル
CZ-6CR1 標準価格4,500円・税別

●ディスプレイテレビ/CZ-6TU用テレビコントロールケーブル
CZ-6CT1 標準価格5,500円・税別

●SCSI変換ケーブル CZ-6CS1 標準価格12,000円・税別

資料請求券
CZソフト
Oh!X
2体

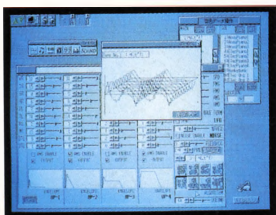
開いてください、触れてくださいウィンドウ、インテリジェンス。
さらに拡がる、SXワールド。

●多彩なサウンドクリエイトを実現するFM音源サウンドエディタ。

SOUND SX-68K

CZ-275MWD 標準価格15,800円(税別)

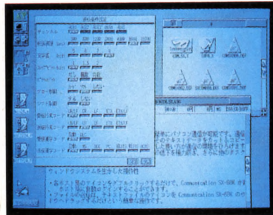
他のミュージックソフトで演奏中の音色を、簡単に作成・変更ができるマルチタスク機能、またエディット、イメージ、ウェーブの3つの編集/確認モードを装備。作成中の音色も50曲の自動演奏でリアルタイムに確認、編集できます。まさにミキサー感覚で音創りが楽しめるツールです。

(2MB、ver1.1)

●マルチタスク機能をはじめ、通信環境がさらに充実。

Communication SX-68K

CZ-272CWD 標準価格19,800円(税別)

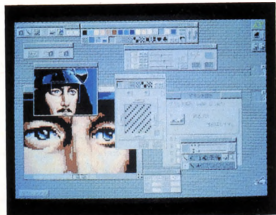
通信環境をさらに高めたウィンドウ対応の通信ソフトです。マルチタスク機能により他のアプリケーションソフトを実行中でも簡単に通信が可能。また、ホスト局をクリックするだけの自動ログイン機能、初心者にも簡単なプログラム機能、最新モデム(20種類)もフルサポートしています。

(2MB、ver1.1)

●ウィンドウ対応グラフィックツール。

Easypaint SX-68K

CZ-263GWD 標準価格12,800円(税別)

マウスによる簡単操作、65,536色中16色の多彩な表現、クリエイティブマインドに応えるウィンドウ対応ペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間でのデータ交換もできます。

(2MB、ver1.1)

●アウトラインフォント対応、ひらかれたウィンドウ環境。

SX-WINDOW ver2.0

CZ-287SS 標準価格12,800円(税別)

フォントマネージャを装備してアウトラインフォントに対応、画面スクロール機能によるワイドデスクトップをはじめ便利機能を満載。 (2MB)

※SX-WINDOW ver1.0およびSX-WINDOW ver1.1をお持ちのかたには有償バージョンアップを行います。

●「SX-WINDOW開発キット」のサポートツール

開発キット用ツール集

CZ-289TWD 開発中

SX-WINDOW開発キットをさらに使いやすくするためのツールです。SXコールのリファレンスを収めた「インサイドSX」、コードリソース作成のためのコンバータ「ハイパーCV」、アプリケーションのインストールが簡単に行える「インストーラ」をはじめ12種のツールが用意されています。 (2MB、ver2.0)

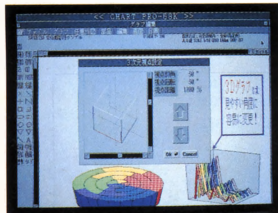
※(2MB、ver1.1)の表示は、メインメモリ2MB以上、SX-WINDOW ver1.1以上が必要であることを示します。

充実のPROシリーズ

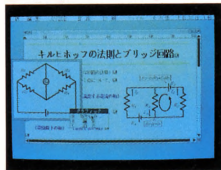
●ビジネスグラフチャート

CHART PRO-68K

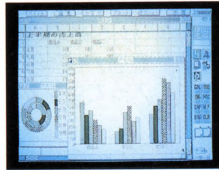
CZ-267BSD 標準価格38,000円(税別)

各種データベースで作成したデータをもとに、多彩なグラフが作成できます。3次元表示やグラフの複合機能も装備。データはMultiword,Press Conductor PRO-68Kに取り込むこともできます。

●グラフィック機能搭載の本格派ワープロ

Multiword ver1.1

CZ-225BSD 標準価格32,000円(税別)

●簡単操作の統合型表計算ソフト

BUSINESS PRO-68K Popular

CZ-286BSD 標準価格28,000円(税別)

●各種ドライバ、ライブラリを追加

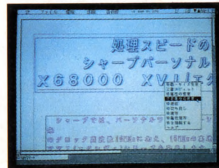
C COMPILER PRO-68K ver2.1

CZ-285LSD 標準価格44,800円(税別)

※有償バージョンアップ対応中。

●各種エディタ装備のレイアウトソフト

PressConductor PRO-68K

CZ-266BSD 標準価格28,000円(税別)

※以上のPROシリーズのソフトの動作にはメインメモリ2MB必要です。

※発売予定のソフトの画面写真は実物とは異なる場合があります。

《業界No.1の"P&Aメンテナンスサポート》

最高の保証システム

①業界最長の新品パソコン5年保証
（※モニター・プリンター3年間保証!! ※一部商品は除きます。）
②中古パソコンの1年間保証
（モニター・プリンター6ヶ月間保証）
③初期不良交換期間3ヶ月
（※新品商品に限らせていただきます）
④永久買取保証
⑤配達の指定OK!!
⑥夜間配送もOK!!
（※PM6:00～PM8:00の間 ※一部地域は除きます。）

便利でお得な支払いシステム

①翌月一括払い手数料無料（ご利用下さい。）
②業界No.1の低金利
③月々の支払いは¥1,000より
④9ヶ月先からのスキップ払いOK!!
⑤84回までの分割、ボーナス併用OK!!
⑥カレッジクレジット
⑦ステップアップクレジット
⑧ボーナスだけで10回払いOK!!
⑨現金一括払いOK!!
（※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。）

1/18～2/17

《増設メモリー&数値演算プロセッサ》計測技研

1 PRKII-02(2M)……定価¥ 55,000▶特価¥ 39,800
2 PRKII-04(4M)……定価¥ 90,000▶特価¥ 67,000
3 PRKII-06(6M)……定価¥125,000▶特価¥ 92,500
4 PRKII-08(8M)……定価¥160,000▶特価¥119,000
5 PRKII-12(2M)……定価¥ 85,000▶特価¥ 63,000
6 PRKII-14(4M)……定価¥120,000▶特価¥ 89,500
7 PRKII-16(6M)……定価¥155,000▶特価¥114,500
8 PRKII-18(8M)……定価¥190,000▶特価¥141,000
9 MC-68881RC……定価¥ 38,000▶特価¥ 27,000

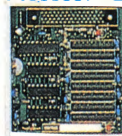
■Z,s STAFF PRO 68K Ver3.0
（ツアイト）（定価¥58,000）
特価¥37,500
（送料・消費税込み¥39,140）

■SX-68M II MIDI
（システムサコム）（定価¥19,800）
特価¥13,500
（送料・消費税込み¥14,420）

■CZ-68HA
●674C用内蔵HD80M
特価~~¥95,000~~
TEL下さい。!!

カラーイメージジェット
■IO-735X-B
定価¥248,000
特価¥152,000
（送料・消費税込み¥157,590）

FDD(5インチ×2基)
■CZ-6FD5
(シャープ)(定価¥99,800)
P&A超特価!!
TEL下さい。

X68000メモリボード

①SH-6BE1-1M(600C専用)(I/Oデータ)……定価¥25,000
（送料・消費税込み¥18,952）……特価¥17,900
②1MB増設RAMボード(ACE/PRO/PRO II用)‥定価¥25,000
（送料・消費税込み¥16,892）……特価¥15,900
③2MB増設RAMボード(拡張スロット用)……定価¥50,000
（送料・消費税込み¥33,166）……特価¥31,700
④4MB増設RAMボード(拡張スロット用)……定価¥88,000
（送料・消費税込み¥57,371）……特価¥55,200

注目!!平成5年4月末一括払い手数料（金利）無料（平成5年2月末/3月末/4月末のいずれかをご指定下さい。）

X68000 Compact XVI/XVI

送料¥3,000、消費税別（クレジット表：送料、消費税込み）

Compact XVI	XVI	XVI-HD	※本体、モニターの組合せも超特価中 TEL下さい。
① ●CZ-674C-H(本体) ●CZ-608D-H(モニター) ●CZ-6FD5(5" FDD) 定価¥492,600 P&A超特価¥285,000	① ●CZ-634C-TN(本体) 定価¥462,800 P&A超特価¥278,000	① ●CZ-644C-TN(本体) ●CZ-608D-H(モニター) 定価¥612,800 P&A超特価¥389,000	左記セットでお買い上げの方にもれなくプレゼント! ①ディスケット10枚、ゲームソフト1ヶはもちろん、さらにその上、人気の ㋑オーバーティク(¥9,800) ㋺ロードス島戦記II(¥9,800) ㋩三国志III(¥14,800) ㊁デスブレイド(¥9,800) ㋭エトワールプリンセス(¥9,800) の中のいずれか1本をプレゼント!!
12回 26,000 24回 13,700 36回 9,500 48回 7,400	12回 24,600 24回 13,000 36回 9,000 48回 7,100	12回 34,400 24回 18,200 36回 12,600 48回 9,900	
上記のモニターをCZ-614Dに変更	上記のモニターをCZ-614Dに変更	上記のモニターをCZ-614Dに変更	
② ●CZ-674C-H(本体) ●CZ-614D-TN(モニター) ●CZ-6CR1(RGBケーブル) ●CZ-6CT1(TVコントロール) ●CZ-6FD5(5" FDD) 定価¥542,800 P&A超特価¥318,000	② ●CZ-634C-TN(本体) P&A超特価¥299,000	② ●CZ-644C-TN(本体) ●CZ-614D-TN(モニター) 超特価¥415,000	左記①のモニターを ①CZ-606D(定価¥79,800)に変更の場合(マイナス)¥9,000 ②CZ-607D(定価¥99,800)に変更の場合¥3,000 ③CU-21HD(定価¥148,000)に変更の場合¥33,000 を加算して下さい。
12回 29,000 24回 15,300 36回 10,600 48回 8,300	12回 26,500 24回 14,000 36回 9,700 48回 7,600	12回 36,700 24回 19,400 36回 13,400 48回 10,500	

X68000シリーズ～P&Aスペシャルセット

（送料¥2,000・消費税別）

SUPER-HD
(CZ-623C-TN)
●ハードディスク81
●平均アクセスタイ
●SCSIインターフェイス標準装備
●SX-WINDOW Ver.1.0搭載
●メインメモリ 2MB標準

SUPER-HD P&A特選セット ★ハードディスク81MB搭載!!

セット	構成	定価	特価
Ⓐセット	■CZ-623C-TN(単品)	定価¥498,000	特価¥178,000
Ⓑセット	■CZ-623C-TN+CZ-606D	定価¥577,800	特価¥233,000
Ⓒセット	■CZ-623C-TN+CZ-608D	定価¥592,800	特価¥246,000
Ⓓセット	■CZ-623C-TN+CZ-607D	定価¥597,800	特価¥248,000
Ⓔセット	■CZ-623C-TN+CZ-614D	定価¥633,000	特価¥268,000
Ⓕセット	■CZ-623C-TN+CU-21HD	定価¥646,000	特価¥278,000

注目 スペシャルプレゼント

※ディスケット 10枚
ゲームソフト 1ヶ
プレゼント

ズバリ価格で大奉仕中

PRO-II P&A特選セット 限定

セット	構成	定価	特価
Ⓐセット	■CZ-653C(単品)	定価¥285,000	特価¥129,000
Ⓑセット	■CZ-653C+CZ-606D	定価¥364,800	特価¥186,000
Ⓒセット	■CZ-653C+CZ-604D	定価¥379,800	特価¥188,000
Ⓓセット	■CZ-653C+CZ-608D	定価¥379,800	特価¥198,000
Ⓔセット	■CZ-653C+CZ-607D	定価¥384,800	特価¥200,000
Ⓕセット	■CZ-653C+CZ-614D	定価¥420,000	特価¥220,000
Ⓖセット	■CZ-653C+CU-21HD	定価¥433,000	特価¥230,000

X68000用ハードディスク（送料¥1,000 消費税別）

〈ロジテック〉
①LHD-FM100E(定価¥99,800)
▶P&A超特価TEL下さい。
②LHD-FM200E(定価¥138,000)
▶P&A超特価TEL下さい。

〈エニックス〉
③EFX-100(定価¥118,000)
▶P&A超特価TEL下さい。
④EFX-140(定価¥138,000)
▶P&A超特価TEL下さい。

〈システムサコム〉
⑤HD-J100(定価¥128,000)
▶特価¥61,000
⑥HD-J170(定価¥189,000)
▶特価¥89,500

〈ジェフ〉
⑦GF-120 定価(¥108,000) 特価¥70,000
⑧GF-200 定価(¥138,000) 特価¥89,000
⑨GF-240 定価(¥148,000) 特価¥98,000

プリンター（送料¥1,000 消費税別）

（ケーブル 用紙 付）

■CZ-8PC5-BK
（定価¥96,800）
▶特価¥68,500

■CZ-8PK10
（定価¥97,800）
▶特価¥71,000

モデム

■PV-M24B5
（AIWA）（定価¥39,800）
▶特価¥25,000
（送料・消費税込み¥26,780）

■MD-24FB5V
（オムロン）（定価¥39,800）
▶特価¥25,500
（送料・消費税込み¥27,295）

■FMMD-311G
（富士通）（定価¥35,800）
▶特価¥24,800
（送料・消費税込み¥26,574）

P&A特選パソコンラック（消費税別）（送料無料）

①3段¥8,900 ②4段¥9,900 ③5段¥12,500

1230(H)×600(D)×650(W) 消費税込み¥9,167
1250(H)×700(D)×640(W) 消費税込み¥10,197
1310(H)×700(D)×640(W) 消費税込み¥12,875

●全機種=移動自由(キャスター付) ●コードクランプ付(4段/5段) ※5段のみ=電源コード付(2.5m)(2P) キーボード収納可能

●本広告の掲載の商品の価格については、消費税は含まれておりません。 ●営業時間=平日AM10:00～PM7:00、日祭AM10:00～PM6:00

アフターサービス万全
全商品保証付。専門の担当がお客様の立場で対応します。
初期不良、輸送トラブルetc.
万が一初期不良、輸送トラブルが発生しました際には、即交換させていただきます。

★頭金なし!!
★即日発送!!

秋 でおなじみのP&Aがズバリ超特価セールでご奉仕!!

- ●お近くの方は、お立寄下さい。専門係員が説明いたします。
- ●本体単品でも受付します。詳しくは、お電話にてお問合せ下さい。
- ●ビジネスソフト定価の15%引きOK!! TEL下さい。
- ●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せ下さい。

全国通販

X68000用ソフトコーナー (送料1ヶ～5ヶまで¥500・消費税別)

商品	定価	特価
◆Z's STAFF PRO68 Ver3.0(ツアイト)	定価¥58,000	特価¥37,500
◆Z's TRIPHONY デジタルクラフト(ツアイト)	定価¥39,800	特価¥27,000
◆テラッツォ(ハミングバード)	定価¥19,400	特価¥13,600
◆マジックパレット(ミュージカルプラン)	定価¥19,800	特価¥14,200
◆たーみのる2(SPS)	定価¥17,800	特価¥13,000
◆Mu-1 Super	定価¥39,800	特価¥28,500
◆CMA68K(シティソフト)	定価¥29,800	特価¥21,800
◆サイクロンEXPRESSα68	定価¥98,000	特価¥69,000
◆C-TRACE68 Ver3.0(キャスト)	定価¥98,000	特価¥68,500
◆G68K Ver.2 PRO	定価¥22,000	特価¥17,300
◆C&Professional Pack V3.2(マイクロウェアジャパン)	定価¥80,000	特価¥57,800
◆ウエットペイント1～3(ウエーブトレイン)〔各〕	定価¥15,000	特価¥11,500
◆マチエール(サンワード)	定価¥39,800	特価¥28,800
◆Windex PRO68(JEL)	定価¥28,000	特価¥20,500
◆CZ-213MSD MUSIC PRO68K	定価¥18,800	特価¥13,200
◆CZ-214MSD SOUND PRO68K	定価¥15,800	特価¥11,300
◆CZ-215MSD Sampling PRO68K	定価¥17,800	特価¥12,500
◆CZ-220BSD DATA PRO68K	定価¥58,000	特価¥40,000
◆CZ-224LSD The 福袋 Ver2.0	定価¥ 9,900	特価¥ 7,400
◆CZ-225BSD Multiword Ver1.1	定価¥32,000	特価¥23,000
◆CZ-243BSD CYBERNOTE PRO68K	定価¥19,800	特価¥15,000
◆CZ-247MSD MUSIC PRO68K〔MIDI〕	定価¥28,800	特価¥20,500
◆CZ-249GSD CANVAS PRO68K	定価¥29,800	特価¥22,000
◆CZ-251BSD Hyper word	定価¥39,800	特価¥29,400
◆CZ-253BSD CARD PRO68K Ver2.0	定価¥29,800	特価¥22,700
◆CZ-257CSD Communication PRO68K Ver2	定価¥19,800	特価¥15,300
◆CZ-258BSD Teleportion PRO68K	定価¥22,800	特価¥16,900
◆CZ-261MSD MUSIC studio PRO68K Ver2.0	定価¥28,800	特価¥21,200
◆CZ-263GWD Easypaint SX-68K	定価¥12,800	特価¥ 9,800
◆CZ-265HSD New PrintShop Ver2.0	定価¥20,000	特価¥15,400
◆CZ-266BSD PressConductor PRO68K	定価¥28,800	特価¥22,000
◆CZ-267BSD CHART PRO68K	定価¥38,000	特価¥29,800
◆CZ-284SSD OS-9/X68000 Ver2.4	定価¥35,800	特価¥25,600
◆CZ-285LSD C-Compiler PRO68K Ver2.1	定価¥44,800	特価¥32,500
◆CZ-286BSD BUSINESS PRO68K Popular	定価¥28,000	特価¥20,500
◆CZ-287SS SX-WINDOW Ver2.0	定価¥12,800	特価¥ 9,800

★ゲームソフト25%OFF!!(一部ソフト除く)

周辺機器コーナー (送料¥500・消費税別)

商品	定価	特価
①CZ-8NS1	定価¥188,000	特価¥133,000
②CZ-6VT1	定価¥ 69,800	特価¥ 49,500
③CZ-6TU	定価¥ 33,100	特価¥ 23,900
④BF-68PRO	定価¥ 19,800	特価¥ 14,400
⑤CZ-8NM3	定価¥ 9,800	特価¥ 7,200
⑥CZ-8NT1	定価¥ 13,800	特価¥ 10,000
⑦CZ-6BE2A	定価¥ 59,800	特価¥ 42,800
⑧CZ-6BE2B	定価¥ 54,800	特価¥ 39,300
⑨CZ-6BE2D	定価¥ 54,800	特価¥ 39,300
⑩CZ-6BF1	定価¥ 49,800	特価¥ 35,800
⑪CZ-6BP1	定価¥ 79,800	特価¥ 57,000
⑫CZ-6BM1	定価¥ 26,800	特価¥ 19,300
⑬CZ-6EB1	定価¥ 88,000	特価¥ 63,000
⑭AN-S100	定価¥ 36,600	特価¥ 26,300
⑮CZ-6SD1	定価¥ 44,800	特価¥ 32,500
⑯CZ-6BN1	定価¥ 29,800	特価¥ 21,500
⑰CZ-6BV1	定価¥ 21,000	特価¥ 15,200
⑱CZ-6BC1	定価¥ 79,800	特価¥ 57,000
⑲CZ-6BG1	定価¥ 59,800	特価¥ 43,000
⑳CZ-6BU1	定価¥ 39,800	特価¥ 28,500
㉑CZ-6PV1	定価¥198,000	特価¥142,000
㉒CZ-6BS1	定価¥ 29,800	特価¥ 21,500
㉓CZ-8NJ2	定価¥ 23,800	特価¥ 17,500
㉔CZ-6BL2	定価¥298,000	特価¥214,000
㉕JX-100S	定価¥ 89,800	特価¥ 44,000
㉖JX-220X	定価¥168,000	特価¥121,000
㉗IO-735XB	定価¥248,000	特価¥152,000
㉘LC-10C1H	定価¥598,000	特価¥459,000
㉙CZ-6CS1(674C用)	定価¥ 12,000	特価¥ 8,900
㉚CZ-6CR1(RGBケーブル)	定価¥ 4,500	特価¥ 3,600
㉛CZ-6CT1(テレビ・コントロール)	定価¥ 5,500	特価¥ 4,400
㉜CZ-6BP2	定価¥ 45,800	特価¥ 33,300

P&A特選＝今月の中古特選品

●CZ-601C
●CZ-611D-TN
¥120,000

限定

●CZ-634C-TN
●CZ-606D-TN
¥198,000

限定

●CZ-644C-TN
●CZ-604D-TN
¥298,000

買取り価格

機種	価格	機種	価格
●CZ-634C	¥150,000	●CZ-602C	¥ 68,000
●CZ-644C	¥200,000	●CZ-612C	¥ 78,000
●CZ-604C	¥ 80,000	●CZ-652C	¥ 48,000
●CZ-623C	¥110,000	●CZ-662C	¥ 68,000
●CZ-603C	¥ 78,000	●CZ-611C	¥ 58,000
●CZ-613C	¥ 90,000	●CZ-601C	¥ 45,000
●CZ-653C	¥ 68,000	●CZ-674C	¥150,000
●CZ-663C	¥ 75,000		

下取り交換差額表

新品 / 下取り	CZ-634C モニターセット	CZ-644C モニターセット	モデル UX20セット	モデル CX20セット	9801FA2
CZ-623C モニターセット	150,000	270,000	70,000	160,000	140,000
CZ-613C モニターセット	190,000	290,000	100,000	190,000	170,000
CZ-652C モニターセット	230,000	340,000	150,000	240,000	190,000
CZ-604C モニターセット	180,000	290,000	100,000	190,000	150,000
CZ-600C モニターセット	230,000	340,000	150,000	240,000	200,000

注目!!平成5年4月末一括払い手数料(金利)無料(平成5年2月末/3月末/4月末のいずれかをご指定下さい。)

中古・高価現金買取り 下取りOK!!

■まずはお電話下さい。
下取り専用買取り電話▶ ☎03-3651-1884 FAX. 03-3651-0141
■下取り・買取りで、お急ぎの方は、直接当社に来店、または宅急便にてお送り下さい。

買取り価格…完動品・箱/マニュアル/付属品付の価格です。

- ●下取りの場合……価格は常に変動していますので査定額をお電話で確認して下さい。(差額は、P&A超低金利クレジットをご利用下さい。)
- ●買取りの場合……現品が着き次第、2日以内に買取り金額を連絡し、振込み、又は書留でお送り致します。
- ●近郊の方は、P&A本店まで、直接お持ち下さい。即金にて、¥1,000,000までお支払い致します。

- ●最新の在庫情報・価格はお電話にてお問い合せください。
- ●買い取りのみ、または、中古品どうしの交換も致します。詳しくは電話にて、お問い合せ下さい。
- ●価格は変動する場合もございますので、ご注文の際には必ず在庫をご確認下さい。
- ●本商品の掲載の価格については、消費税は、含まれておりません。
- ●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せ下さい。

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで、現金書留でお送りください。(プリンター・フロッピーの場合、本体使用機種名を必ず記入のこと)

〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。
(電信扱いでお振込み下さい。)

〔振込先〕さくら銀行 新小岩支店
当座預金 2408626 ㈱ピー・アンド・エー

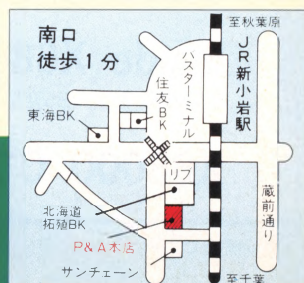
〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。
●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。
●1回～84回払いまで出来ます。但し、1回のお支払い額は¥1000円以上。

《便利な超低金利クレジットをご利用下さい》

- ●月々¥1,000円からOK!!
- ●ボーナス払いOK(夏冬10回までOK)
- ●支払い回数 1回～84回
- ●お支払いは、8ヶ月先からでもOK!!

超低金利クレジット率

回数	3	6	10	12	15	24	36	48	60	72
手数料	3.0	4.0	5.5	5.5	8.5	11.5	16.0	21.0	27.0	33.0

●定休日/毎週水曜日

マイコン専門ショップ P&A
株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目1番地19号
☎03-3651-0148(代) FAX. 03-3651-0141

営業時間
平日:AM10:00～PM7:00
日祭:AM10:00～PM6:00

●価格は流通事情により変動致しますので、銀行振込・書留等の送付前に、あらかじめお電話にてご確認下さい。

2月号

1月18日(月)発売

特別定価
720円(税込)

パーソナルコンピュータ総合情報誌

MONTHLY SPECIAL

98、Mac、TOWNS、DOS/Vに
新モデルぞくぞく誕生

1993年の買いは CD-ROMパソコンだ

BEST BUY '92

- 速くて安い486/66パソコンのベストマシンは?
- これからパソコン通信をする人のためのはじめてのポケットモデムは?
- 自宅で活用する、わかりやすい表計算ソフトのベストセレクトは?

特別企画

——98、Mac、TOWNSをもっと楽しく使う

CD-ROMソフトのレビューと一覧

好評連載

- MORE REVIEWS
- パソコンAV塾
- HARD TUNEUP!
- SOFT TUNEUP!
- U.S.A. RUPO
- PC DATA
- COLUMNS

PC SYSTEM UP!

Power Bookで作る
ポータブル・サウンドシステム

新春
特別付録

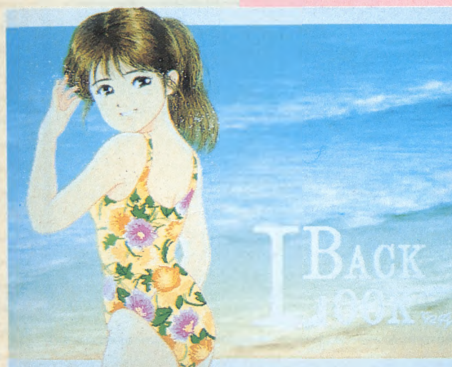
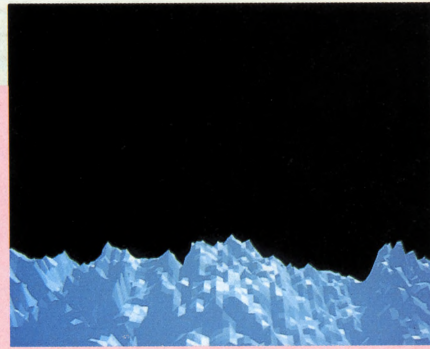
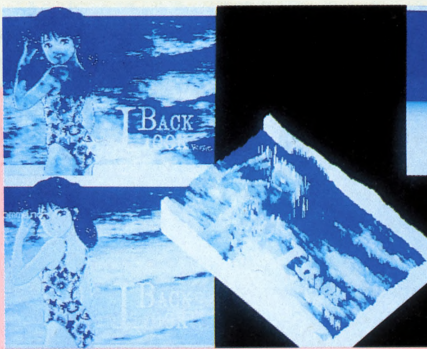
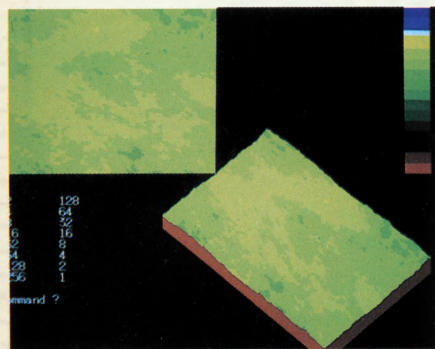
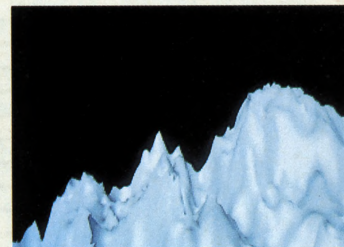
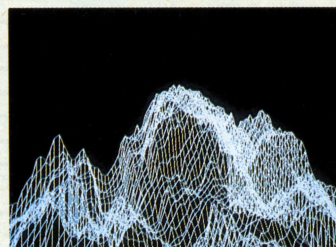
月刊PCオリジナル
福袋ディスク

- DOS版オリジナルパフォーマンステストVer.1
- 中尊寺ゆつこの壁紙

ソフトバンク出版事業部
〒108 東京都港区高輪2-19-13 NS高輪ビル
TEL：03-5488-1360

画像創造のために

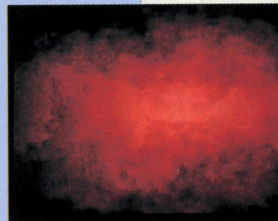
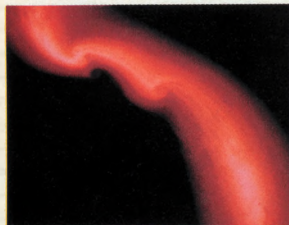
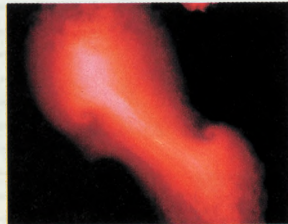
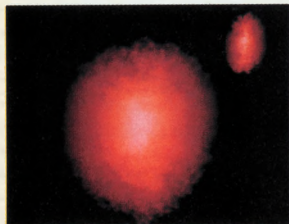
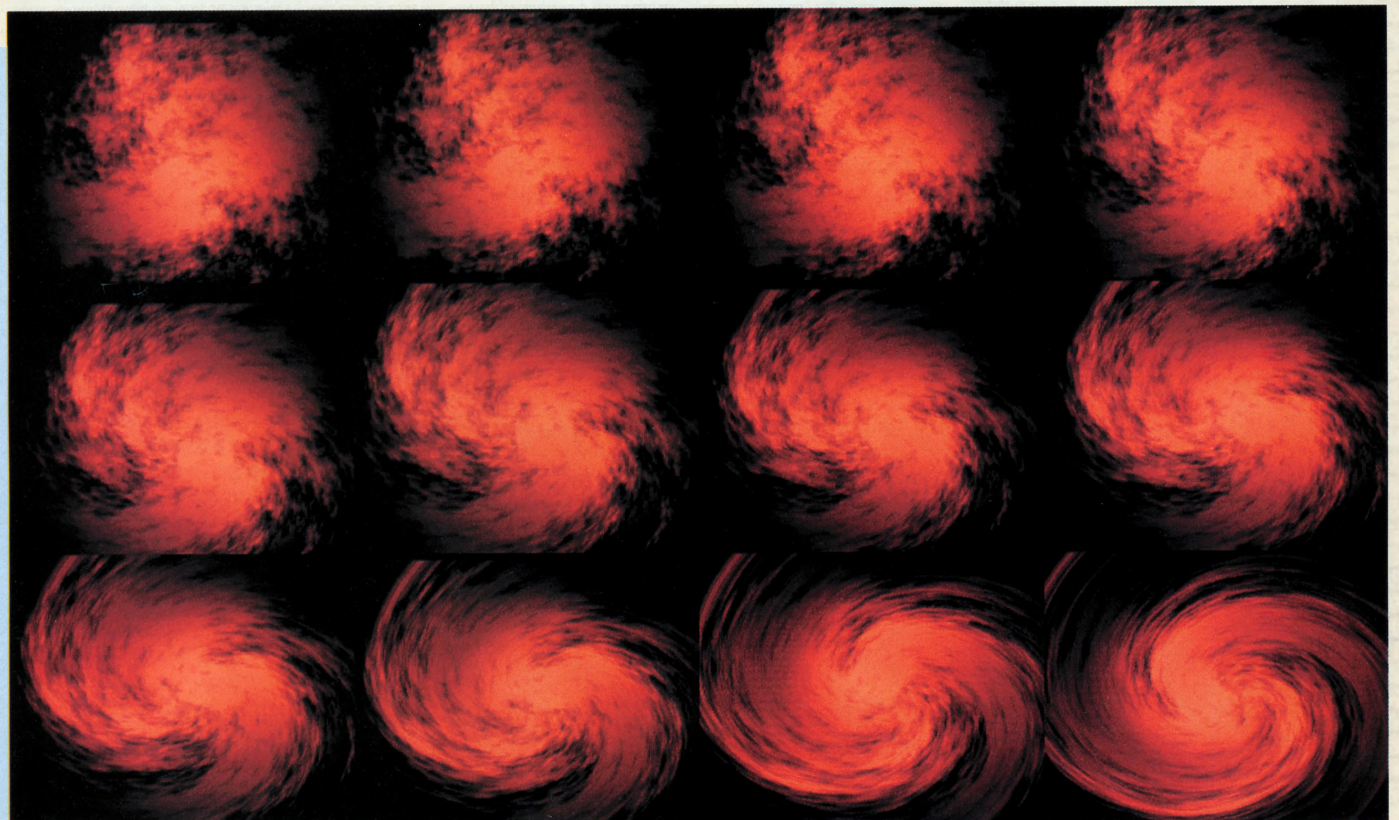
画像/形状データの自動生成を目指して，さまざまな物体/現象についてコンピュータでシミュレートしてみました。従来の方法では困難な自然物を表現してみましょう。

1992年12月号で発表したアルゴリズム（の改良版）で作成した地形データをDōGA CGAシステムを使ってレンダリング/表示してみた。下はもっとも基本的な出力データをそのままレンダリングしたもの。右側の3点は地形の一部を再帰的に合成してメリハリをつけてみたもの（倍率は4倍で合成）。少々メリハリがつきすぎた感じがある。なお異様に尖った部分が見えるのはデータが端の部分で切れているため。
データは127×127ポイントで出力。かなり大きなデータ（1Mバイト程度）だが，近くでは十分な解像度ではない。念のためスムースシェーディングもかけてみたが，滑らかすぎると逆に地形っぽくなくなることがわかる。細かい部分にこそフラクタル処理が必要なのかもしれない。

今回の拡張によって，生成される地形を（うまくやれば）ユーザーが制御できるようになった。あいにく適当なデータがないのでまったく関係ないグラフィックデータを地形とみなして合成するという暴挙に出てみた。
左上が基本地形。左下が変換元前のグラフィックデータ。中央上はそれらを合成したもの。単純な合成では疑似立体表示でも大きな段差を示す。中央下はそれに1段階の平滑化を入れたもの。上はそれを実際に地形とみなしてレンダリングしてみたもの。まあ，あまり期待していたわけではないが，原画の雰囲気がなにもわからない。当たり前と思うか，やり方が悪いと思うか……。

原画像

フィルタの例

1枚絵ではなんだかわからない陽炎フィルタの出力例。画面端の処理を行っていないのでゴミが出ているし，いかにも粗い。それでも動かせばある程度の効果は期待できる。

乱数と再帰処理の組み合わせによって記述した樹木の例。もちろん，これら以外にも工夫しだいでさまざまな形態のものが生成可能だ。X-BASICで作成された関数群を使うことで，DōGA CGAシステム上で3次元タートルグラフィックのようなフレームデータを生成することができる。

渦による空間の変形で表現された煙。制御点が3，5，10と増えるに伴い（位置はランダム），それが流れの強さを示すかのように変形されていくことがわかる。ちょっと渦が強すぎたか？　下は渦の基本動作を実験的に確かめてみたもの。

AMIGAによる画像例

Oh!XでなぜAMIGA？ と疑問を持たれる人も多いかもしれない。

AMIGAシリーズは非常に安価であるにもかかわらず，そこそこの性能で（必要なら性能はある程度上げられるのだ），最先端の環境とまではいかなくても贅沢をいわなければ実用とするには十分な品質のソフトウェアが揃っているのだ。他機種ではハードウェアの値段はかなり下がってはいるものの，ソフトウェアの価格が非常に高い。AMIGAの場合はソフトウェアの値段も極端に安い。一般大衆レベルでも手の出せる唯一のCGマシンといっていいだろう。はっきりいって普通の人にはそんなに高速なマシンは必要ないのだ。

低価格化が進んでAMIGAのようなマシンが手軽に入手できるような環境になりつつある現在，パソコンとしてもこのマシンを無視することはできないだろう。ことグラフィックに関するかぎりAMIGAの存在は脅威ですらある。注目して損はないだろう。

そのAMIGAで使用できるソフトウェアから選んだのがこの2種のツールである。時間さえかければこれだけのクオリティで手軽に出力できるのだから他機種もうかうかできないだろう。最近はMacintoshのCGが話題になることも多いが，しょせんはパソコンの環境という感じである。AMIGAのツールを見ていると「この値段でどうしてこんなことができるんだ？」と感じさせられるものが多い。つくづくソフトウェア技術のベース格差を思い知らされる（他機種というのはグラフィックワークステーションのことである。念のため）。

上の写真はScenery Animatorの出力画像だ。ちょっと「綺麗すぎる」のでCGだとわかるという例だ。被写界深度の設定と微妙な空気遠近法を加えればリアルさが加わるだろう。このあたり一帯が生成されているので内部を自由に動き回れる。ポリゴンの簡易表示にすればほとんどリアルタイムで移動できる。手前の木が欠けているのはバグかもしれない。

下はそのライバルというべきVISTA PROの画像例である。

VISTA PROの画像出力例。Scenery Animatorが乱数による生成という感じで画像を作っていたのに対し，VISTAはもともと「地図や写真で取り込んだ地形を立体化します」というノリで売り出されていたものだ。確か「ロケーションしなくても観光地のパンフレットが作れます」というのが売り文句だった。実際，グラフィックツールでエディットできる。木を植えたいところにブラシをひと吹きすればそこに林ができるという感じで操作できるのだ。

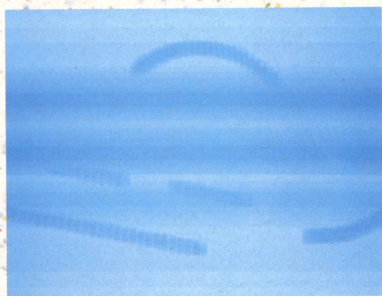
Oh!X

Graphic Gallery

DōGA CGアニメーション講座

CGAマガジンでは自動的に各種アニメーションを作成することができます。しかし，初心者以外の方はオブジェクトや新しいツールを利用して，積極的に作品を作ってもらいたいと思います。今月の連載記事での具体例も参考になることでしょう。

（1フレームおきに撮影）シケインを走り抜けるF1 2台を固定カメラで追ったシーン。左に曲がる場面と右に曲がる場面の2つのカットを作成して，あとからつなぎあわせます。

路面のマッピングデータも収録。

周辺にオブジェクトを置いて，路面にはスリップマークをマッピングしたカット。

[1992年度]

GAME OF THE YEAR

◆ノミネート作品発表◆

編集協力：浦川 博之

選択応募部門

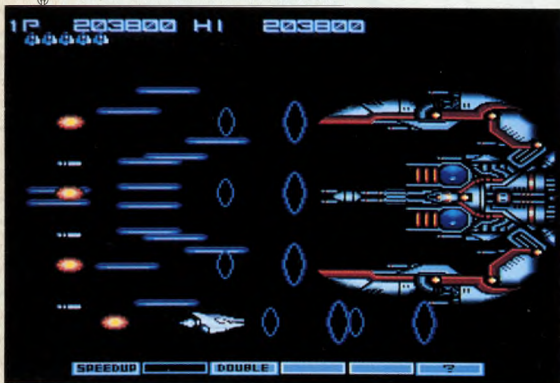
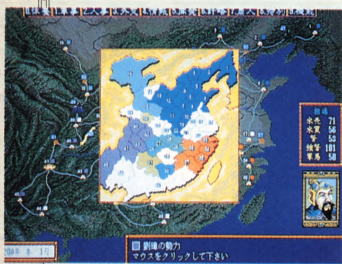
Oh!X ゲーム大賞

いよいよ，1992年度のGAME OF THE YEAR，Oh!Xゲーム大賞を選ぶときがやってきました。このOh!Xゲーム大賞は，数ある名作，傑作のなかでも「こいつがチャンピオン！」という1本にのみ与えられる賞，いわば今年のキングオブゲームです。

今年のOh!Xゲーム大賞は有力馬が目白押し。本命不在の混戦模様で，オッズがついたらさぞかし面白いでしょうな。その中であえて本命を挙げるとすればグラディウスIIとファイナルファイトの2本でしょうか。かたやコナミの代表作，かたやカプコンの出世作。どちらもゲームセンターで注目を一手に集めた前歴を持っており，血統十分，ネームバリューも実力も文句なし。

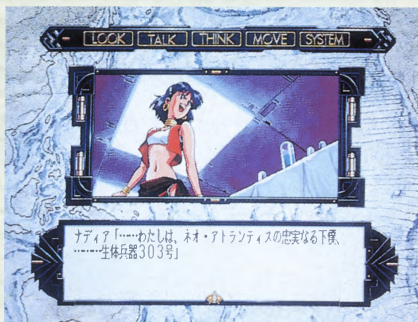
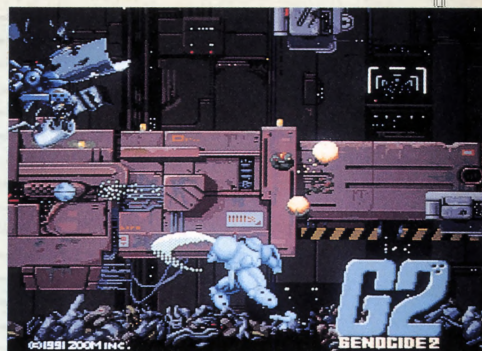
これを追いかけるのがズームのオーバーテイク。評判がまだ固まっていないのと，滑り込みノミネートという不安定要素もありますが，F1ゲームの新鮮さとX68000オリジナルという利点を生かせばトップに踊り出るのはそう難しくないでしょう。

さらに1991年の年末勢も見逃せない存在。スターウォーズ，出たな!! ツインビー，ジェノサイド2といった面々は，昨年度はいまひとつ実力を発揮できなかった面もあり，今回は真価を問うべく再チャレンジ。

今年度はOh!Xゲーム大賞にふさわしい風格のあるゲームが多く，票数的には意外に低いところで決着がつきそうです。ということは，読者の1票がいつも以上に重いということ。あなたが考える代表作はどれか，もう一度じっくり1992年を振り返って「これだ！」と思う作品を選んでみてください。

ノミネート作品

- グラディウスII
- スターウォーズ
- ファイナルファイト
- 出たな!! ツインビー
- ジェノサイド2
- オーバーテイク
- ポピュラスII
- ふしぎの海のナディア
- 三國志III
- 大戦略III'90
- レミングス
- シムアース
- ストライダー飛竜
- テラクレスタ/ムーンクレスタ
- エイリアンシンドローム

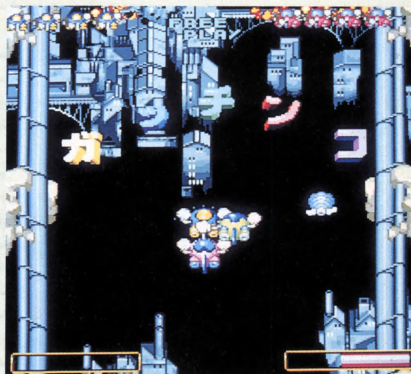
グラフィック賞

「Oh!Xゲーム大賞には入らないけど，このゲームのここんとこの頑張りは評価してあげたい！」というあなたの声のための部門賞。まずはX68000らしさがいちばん出せる分野，グラフィック賞です。グラフィックとなると，やはりX68000オリジナルの作品に注目が集まります。筆頭はなんといってもズーム。統一された画面センスを誇るジェノサイド２には，時期的な不利を跳ね返すパワーがあります。対するバーンウェルトは正当派ジャパニーズアニメの路線で高いクオリティを実現，シュートレンジは，画面の随所に小技を効かせたキモチいいグラフィックで攻めています。突出した存在はないにせよ，X68000のグラフィックパワーを使いこなすメーカーが広がってきたことは嬉しい傾向ですね。

ノミネート作品

- ジェノサイド2
- ふしぎの海のナディア
- シュートレンジ
- バーンウェルト
- エイリアンシンドローム

音楽賞

MIDIもかなり普及して，ユーザーの耳も肥えてきた昨今。ゲーマーを満足させ，ゲームを盛り上げる音楽環境を提供するのはなかなか大変になってきています。そのユーザーの厳しい目と耳を乗り越えたのが下に掲げた５作品です。読者の皆さんも自分の耳で確かめたことでしょう。

カプコンの猛追を受けているズームとコナミも，この分野に関してはやはり一日の長があります。特にオーバーテイクはレースゲームのポイントでありながら，再現が難しいといわれていたレースカーのエグゾーストノートをPCMとFM音源の合わせ技でクリア。従来にないサウンドリアリティを実現しています。グラディウスⅡは，SC-55とMT-32とで対応の仕方を変えるなど，執念すら感じさせるこだわりが支持を受けそう。去年はズームがコナミを抑えましたが，今年は？

ノミネート作品

- オーバーテイク
- ジェノサイド2
- グラディウスⅡ
- 出たな!! ツインビー
- スタートレーダー

1992年度ゲームソフトの傾向と対策

冬です。２月です。今年もGAME OF THE YEARの季節がやってきました。まずはここに出そろった各部門賞のノミネート作品をご覧あれ。これらの作品の中から，あなたの思い入れがこもった１票によって，栄えある受賞作品が選ばれるわけです。

ノミネート作品は，TOP10での人気作品を中心に編集部が調整を加えて作成したラインナップです。いずれも賞の名に恥じないグレードの高いゲームばかり。この中からさらに1992年を代表する作品を選ぶんだから，ワクワクする話でしょ？

振り返ってみれば，今年ほど大作のニュースの絶えない１年はなかったように思います。なにしろ1991年末のアクションが豊作だったのに加えて，春にはグラディウスⅡ，夏にはファイナルファイトが登場。秋になるとズームから待望のオーバーテイクが発売になるという始末。これじゃユーザーの目はアクションゲームに釘付け。しかも従来のシューティングに加えて格闘ものとレースゲームが登場したことで，アクションゲームのバリエーションも増えました。

この余波をくらったというわけでもないでしょうが，アクション以外のゲームはいまひとつ元気がなかったといえるかもしれません。シムアースにレミングス，三國志Ⅲとネームバリューでは，決してアクション勢に負けるものではないのですが，去年のイースやボンバーマンのような存在感のある「名脇役」になれた作品がなかったようです。出来そのものはいいだけに，読者がどのような判定を下すのか注目されます。今年のGAME OF THE YEARの一番の見どころは，やはり王道を行く作品の間のし烈なつばぜりあいにあるといえそうですね。

プログラミング技術賞

1992年は技術力をゲームの土台となる部分に注ぎ込むところが多く，技術を前面に押し出した作品は少なかった1年でした。

といっても，もちろん技術が後退するはずもなく，「ええっ，これがX68000で動くの？」というタイトルが登場して，ユーザーはその恩恵を十分にこうむっているわけです。取捨選択の部分はあったものの，X68000へのゲーム性を損なわない移植の方法については，もはや完全にノウハウが確立した感があります。また，パソコンゲームにもポリゴンによるロボットバトルなど，やや消化不良ながらもコンセプトの進んだ作品が登場しており，ビデオゲーム一辺倒の傾向に逆らって頑張っています。

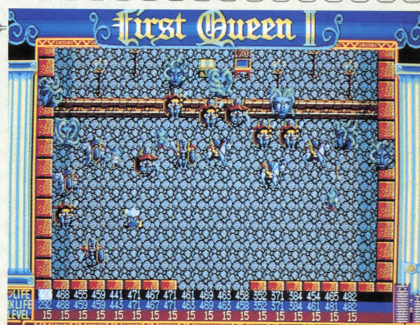
ノミネート作品

オーバーテイク
バトルテック
ファイナルファイト
コード・ゼロ
ストライダー飛竜

ゲームデザイン賞

すぐれたコンセプトを持ち，高いゲーム性を実現していたソフトに贈られるのがこのゲームデザイン賞。昨年度はシンプルなボンバーマンが受賞しており，「ゲームデザイン」という言葉の奥の深さを感じさせられる賞です。コンセプトという面では築城をゲームに仕立てあげてしまったキャッスルズが光ります。ポピュラスⅡやスピンディジーⅡは，インパクトというよりもゲームバランスで勝負といったタイプ。ゲームそのものは，どこでも見られるようなタイプではないうえに，Ⅱとつくだけあって，斬新なシステムやできることをよく追求してある点が魅力です。「遊べる」という点では奥の深さととっつきやすさを両立したレミングスが一歩リードしている感もありますし，今年のゲームデザイン賞の行方はどうなるのでしょうか。

ノミネート作品

レミングス
ファーストクイーンⅡ
キャッスルズ
スピンディジーⅡ
ポピュラスⅡ

西川　善司 <1992>

去年私がよく遊んだのは，出たな!!　ツインビーとグラディウスIIか。美しいグラフィックと3種類の音源に対応したBGM，派手なSE，そして細部に渡ってオリジナルそっくりのゲーム性。ひさびさにわが家のジョイスティックを手汗で濡らした。出たな!!　ツインビーは，アーケードでの登場から間もない発売とあって，興奮もひとしおであった。

あとは，手を出したらハマったというのにファイナルファイトがあった。結局，技の出し方をすべて把握できなかったが，ドカドカ敵をなぎ倒していく快感は，馴染みのないものでとても新鮮であった（私は格闘モノは基本的に食わず嫌いだった）。X68000のソフト市場がピンチとか騒がれるが，X68000の得意分野といわれるアーケードゲームの移植作品のタイトル数はまだまだ多い。出たな!!

ツインビーに始まり，グラディウスII，ファイナルファイト，エイリアンシンドローム，テラクレスタ/ムーンクレスタ，チェイスH.Q.，デスブレイド，ストライダー飛竜などなど。一部を除いていずれも移植忠実度は高いものばかり。各ソフト会社が蓄積したソフトウェア技術はもちろんだが，X68000の大容量メモリ構造とオリジナルの動作環境に似た，マシンスペックのなせる技である。

1993年も，多くのアーケードゲームの移植ものが発売されることを願いたい。電波，コナミ，カプコン，SPSほかのソフトハウスさん，がんばってね～ん。

自由応募部門

主演・助演キャラクター賞

なくてはならないキャラクターがいる。たとえ，ゲーム中に失われる運命にあろうとも……。そんな，ゲームの立役者であるキャラクターたちの中から，もっともゲームに貢献した，もしくはゲームに花を添えたキャラクターを選ぶのが，この主演・助演キャラクター賞です。な〜んて，こう聞くと真面目そうな賞と思うかもしれませんが，過去に受賞したキャラクターたちといえば，テトリスの直線ブロック，サイバースティック，パワーモンガーの羊などなど，結構いろものが多かったりするんです。

まあ，故意にいろもの路線で突っ走るのもいいですが，やっぱり基本を忘れないでください。ただ，変なものじゃあ面白くありませんからね。ゲームに対するキャラクターの位置づけをよく考え，あなたが「これだ！」と思ったキャラクターに投票してください。

底抜け脱線ゲーム体験談

いまだ該当作品の出たことのない「底抜け脱線ゲーム賞」。今年は，ちょっと趣を変えることにしました。遊んでみると結構イイ線いっているな，と思っていたら突然腰砕けになってしまったとか，前評判を信用してソフトを買ったら，ほにゃららな路線へ走っていってしまっていた，というような読者の皆さんが体験した，底抜け脱線ゲームを募集します。いわゆる，ク◯ゲーを見つけたよ，という報告ではなく，本来とは別の意味で面白い，楽しかったゲームの体験談を明るく紹介してください。

また，本来の遊び方とはちょっと脱線した遊び方，ソフトを楽しく遊ぶためのテクニックなどもこのコーナーに含まれます。やっぱりゲームは楽しく遊ばなくきゃね。あなたが見つけた，あなたなりの遊び方を紹介してみませんか。

勝手にGAME OF THE YEAR & 読者レビュー

きっちりノミネートの決まったGAME OF THE YEARですが，やっぱり読者の皆さんにはそれぞれにひいきをしたいゲームがあると思います。そんな思いをぶつける場が，この「勝手にGAME OF THE YEAR」のコーナーです。おちゃらけ賞を設定しようが，勝手にノミネートを設定してアナザーYEARを形成したり……選考者を納得させるか，爆笑させれば勝ちです。読者の皆さんの手で，誌面スペースを勝ち取りましょう。

さて，今年から「ゲーム回顧録」の代わりに読者によるゲームの「読者レビュー」を募集します。対象となる作品は，23ページにあるOh!X1992年度作品リストにあるゲームです。このゲームに関して書きたいことがある！　というような情熱をぶつけてみませんか。いいものがあれば何人でも載せるつもりでいます。がんばって投稿してください。

八重垣　那智 <1992>

1992年を簡単に表現するならば，「大物の年」ということになるだろう。前年とうって変わり，話題作も人気作も集中する傾向が見られた。年始めのころは，前年の話題作集中傾向の影響があったとはいえ，水も温む頃にはグラディウスIIという，今年を象徴する大物が登場している。

硬派のシューティングゲームという，ややプレイヤーを選ぶジャンルでありながら，そのネームバリューは驚異的であり，実際の内容も実に風格のある，ヘビーなものであった。オリジナルが4年前であるにもかかわらず，歯ごたえのある難易度に，指先を熱くした人は少なくないだろう。

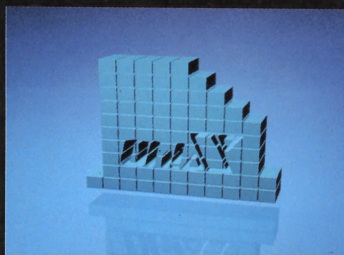
ほかにもファイナルファイトといった，格闘ものに，オーバーテイクといったレースものなど，アクションゲームに関しては，味のある話題作が非常にいいタイミングで交互に出てきたといえるだろう。

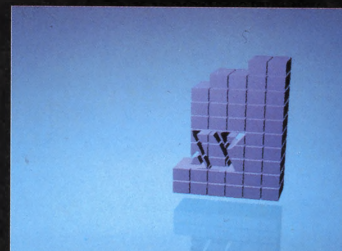
しかし，ミーハーな話題作については，非常に理想的な商品展開をしたわけだが，ある意味で趣味や好みの分かれるような，癖のあるソフトに関して，ややコマ不足といった印象は否定できない。これは，そういった作品の供給源になっている，アーケードゲームに同様な一点集中傾向があるということと無関係ではないだろう。

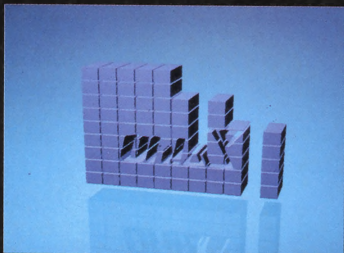
やはり，たまにはRPGやシミュレーションなどをやってみたくもなる。選択の幅は広く，深くあってほしいものである。

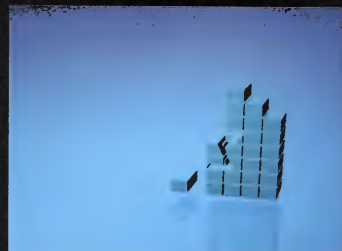
1992年3月～1993年1月までの TOP10総合得点順位

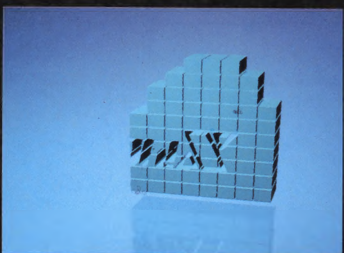
今年も，読者からの人気投票で決まるTOP10のコーナーを集計してみました。1992年は，どんな作品が読者の支持を受けたのでしょうか。

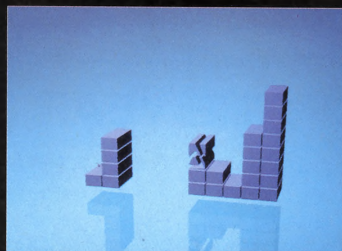
1位　グラディウスⅡ　83

発売と同時に一気にトップへ躍り出て，今年度はほとんど上位にいるという，非情なまでの強さを見せつけている。この強さがそのまま作品の出来に結びついているなら，他のソフトがGⅡの牙城を突き崩すのは難しいかもしれない。このままGⅡの独壇場となってしまうのか？

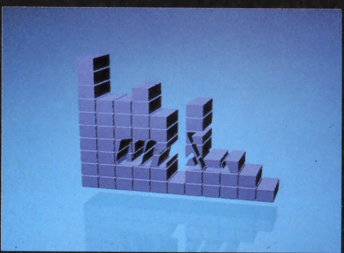
6位　オーバーテイク　48

発売は11月ながらも，前評判でこれだけの人気を得られるのは，さすがX68000ユーザーに定評のあるズームといったところ。ファイナルファイトからトップの座を奪い，これからの動向，GAME OF THE YEARでどういった評価を受けるか，いちばん注目される作品である。

2位　スターウォーズ　72

グラディウスⅡに頭を押さえつけられた感じとなってしまったが，総合得点ではそれほど負けているわけではない。1年たったからといっても，作品のよさは薄れてはいないから，今年度のGAME OF THE YEARでもかなりの健闘ぶりを見せるだろう。

7位　ポピュラスⅡ　26

作品自体はもうしぶんないものであるが，いまひとつ話題性に欠けてしまった感じがある。今年のイマジニアが送り出した作品に共通している欠点といえるかもしれない。他の大作ソフトに埋もれてしまったのだろうか。上位を狙える作品なのは間違いないはずなのに。

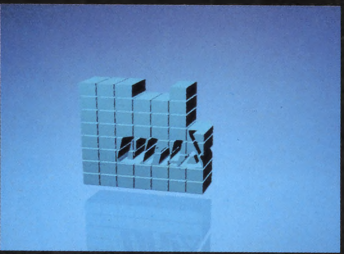
3位　ファイナルファイト　69

移植決定によるアーケードファンからの熱い推薦と，その完成度の高さで3位を獲得。なみいる強豪の中にあって，初参入でこれだけのソフトをぶつけてきてくれたことを，ユーザーは評価したのだろう。次回作という不純な動機も混じっているかもしれないが，手強い存在だ。

8位　ふしぎの海のナディア　25

最初の山は，移植決定の知らせに反応したファンの推薦か？　そのあとしばらく顔を見せないが，本当に発売が近くなってくると，また，盛り上がる。段取りの悪さが票割れの原因といえるが，ファンからの反応は上々なので，GAME OF THE YEARでもいい評価を得られそうだ。

4位　出たな!!　ツインビー　55

GⅡが発売されてから，順位こそ落としているがかなりしつこく食い下がっている。このねばりがGAME OF THE YEARでの評価に直接つながることができるのか。この4位という順位が，作品自体の評価か，それともコナミに対する評価かで順位がきまってくるはず。

9位　三國志Ⅲ　22

発売前後に多少の盛り上がりがあったものの，タイトルのネームバリューに比べ比較的おとなしい得点で終わっている。固定ファンをつかんでいるゲームだけに，ファンからの熱いラブコールによって，大逆転劇となる可能性を秘めているからあなどれない。

5位　ジェノサイド2　51

精細なグラフィック，大胆なアクションと前作以上の頑張りが評価の理由だろう。なかなかいい位置にきているのだが，オーバーテイクの発表とともに順位を下げてしまったのが残念。このズームの2作品で票割れが起こると，かなり不利な立場に追い込まれるだろう。

順位	タイトル	得点
10位	大戦略Ⅲ'90	19
	パロディウスだ！	
12位	レミングス	18
13位	シムアース	16
14位	エトワールプリンセス	14
15位	パワーモンガー	13

10～15位まではどんぐりの背比べといったところ。上位を狙うのは厳しいだろう。作品がきちんと評価されることは間違いないのだが，どのような評価が下されるか注目したい。

〈表の見方〉横軸がTOP10掲載月(1992年3月から1993年1月まで)，縦軸が月間獲得ポイント数(1位＝10ポイント～10位＝1ポイント)です。

Oh!X 1992年度作品リスト

ここでは，Oh!Xが1992年の1年間を通して扱ったゲームの一覧表を掲載します。今年1年でどのような作品が発表されたか，再確認してみてください。

ACT

NIKO²	ウルフ・チーム
ジェノサイド2	ズーム
ヘビーノヴァ	ブラザー工業(TAKERU)
サンダーレスキュー	ブラザー工業(TAKERU)
ストライクレンジ	ブラザー工業(TAKERU)
ファイナルファイト	カプコン
デスブレイド	SPS
バーンウェルト	グローディア
スクエア・リゾート　ハイパー戦車戦	ファミリーソフト
ストライダー飛竜	カプコン

ADV

ふしぎの海のナディア	ガイナックス

RAC

オーバーテイク	ズーム
チェイスH.Q.	ブラザー工業(TAKERU)

PRG

ロードス島戦記　福神漬	ブラザー工業(TAKERU)
ブルトン・レイ シナリオ集 VOL.3	システムソフト
ウルティマVI	ポニーキャニオン
サークII	ブラザー工業(TAKERU)
アルシャーク	ライトスタッフ
ロードス島戦記II－五色の魔竜－	ハミングバードソフト
キングス・ダンジョン	ソフトプラン
ドラゴンナイトIII	エルフ

SHT

出たな!!　ツインビー	コナミ
飛翔鮫	KANEKO
ラストバタリオン	スティング
グラディウスII	コナミ
コード・ゼロ	エニックス
XENON2	エピック・ソニー
スタートレーダー	ブラザー工業(TAKERU)
エイリアンシンドローム	電波新聞社
超人	ブラザー工業(TAKERU)
シューティング68K GAMES	ブラザー工業(TAKERU)
テラクレスタ/ムーンクレスタ	電波新聞社

SLG

ブリッツクリーク	システムソフト
シムアース	イマジニア
大戦略III'90	システムソフト
マスターオブモンスターズII	システムソフト
伊忍道	光栄
ファーストクイーンII	クレソフト
F15ストライクイーグルII　シナリオ集	マイクロプローズジャパン
ロイヤルブラッド	光栄
ノア	M.N.M.Software
ドラゴンストライク	ポニーキャニオン
ライフ&デス	ブラザー工業(TAKERU)
太閤立志伝	光栄
三國志III	光栄
バトルテック	ビクター音楽産業
ライジングサン	ビクター音楽産業
ヨーロッパ戦線	光栄
ポピュラスII	イマジニア
リーディングカンパニー	光栄
ネクタリス	システムソフト
キャッスルズ	ビクター音楽産業
シュートレンジ	ビッツー
エアーマネジメント	光栄
将棋聖天	ホームデータ
棋太平68K	SPS
パチンコ・ワールド	ブラザー工業(TAKERU)
麻雀遊園地	ホームデータ
ディノランド	ブラザー工業(TAKERU)
ワールドゴルフIII	エニックス

PUZ

ユニオン	ポニーテールソフト
PITAPAT	ビクター音楽産業
レミングス	イマジニア
スピンディジーII	アルシスソフトウェア
スーパー上海ドラゴンズアイ	ブラザー工業(TAKERU)
パイプドリーム	BPS

クイズ

苦胃頭捕物帳	電波新聞社

応募方法

ここで，1992年度GAME OF THE YEARへの投票方法について，もう一度詳しく説明させてもらいます。

1)　アンケートハガキを使用する場合

今月号のアンケートハガキには，Oh!Xゲーム大賞の作品名と推薦理由。そして，1.グラフィック賞，2.音楽賞，3.プログラミング技術賞，4.ゲームデザイン賞の作品名と，その4項目のうち1項目について推薦理由を記入してください。自由部門賞については，編集室へのメッセージ欄を使って投票していただいてもかまいません。書く分量があまりにも多くなった場合は，官製ハガキか封書で投票するようにしてください。

2)　官製ハガキまたは封書の場合

まず，宛先は，
Oh!X編集室内
「1992年度GAME OF THE YEAR」係
です。アンケートハガキだけではもの足りない人，「勝手にGAME OF THE YEAR」に投票したい人は，ご自分で官製ハガキ，または封書で投稿してもらうことになります。原稿フォーマットの制限はありませんが，投票したい賞名，作品名，推薦理由がはっきりわかるようにしてください。

また，「勝手にGAME OF THE YEAR」ではゲームに関するイラストも募集しています。サイズの制限，内容の制限は特にありませんが，モノクロでお願いします。

3)　読者レビューの場合

宛先は，
Oh!X編集室内
「GAME OF THE YEAR
読者レビュー」係
です。応募する場合は，「GAME OF THE YEAR」の投票と別にして封書でお送りください。分量は，400字詰め原稿用紙2枚分(800文字)までとします（ディスク可）。ひとりで複数のゲームレビューを書いて応募することもできますが，採用されるものはひとつとなりますので注意してください。

また，例年どおりGAME OF THE YEARのメッセージ採用者から，抽選で謎のプレゼントXを送ることになりますので，ふるってご応募ください。お待ちしています。

Oh!Xゲーム大賞	ゲーム大賞推薦理由
1.グラフィック賞	
2.音楽賞	推薦理由（　）
3.プログラミング技術賞	
4.ゲームデザイン賞	
各賞の作品名を記入	推薦したい賞番号とその推薦理由を記入

SOFTWARE INFORMATION

次は「チェルノブ」！　なかなかスゴイところをついてくるなあ。知らない人にはキワモノ的に捉えられそうだけど，一部ではいまだに人気のあるゲームなのだ。発売が遅れていたソフトもいろいろ出そう。

チェルノブ

「テラクレスタ/ムーンクレスタ」に続く，“ビデオゲーム・アンソロジー”シリーズ第2弾が早くも登場。今度は，データイーストの横スクロール型アクションゲーム，「チェルノブ」が移植される。オリジナルとなるアーケード版は，旧ソ連でのチェルノブイリ発電所事故の記憶も新しい1988年に登場。タイトルやゲーム全体に流れる独特の雰囲気で話題を読んだ。

で，内容はというと，人間発電所“チェルノブ”が数段階のショットを撃ったり，敵を踏み潰したりしながら，悪に立ち向かうべく走り回る。

また，オマケとして，メガドライブのジョイパッドをX68000につなぐためのアダプターを同梱することが予定されている。これなら“「チェルノブ」は3ボタンじゃなきゃやだ”という人も安心だろう。

1月下旬に出る予定なので，“蓄電”しながら発売を待とう。

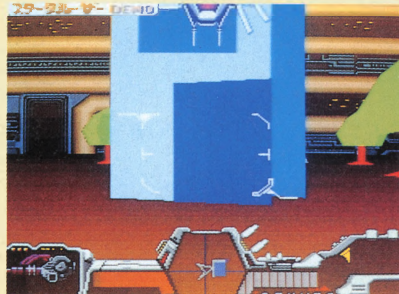
X68000用　5"2HD版　4,900円(税別)
電波新聞社　☎03(3445)6111

3.5インチユーザーに捧ぐ

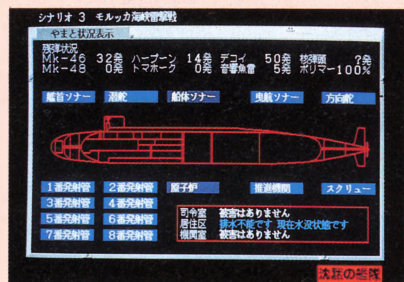
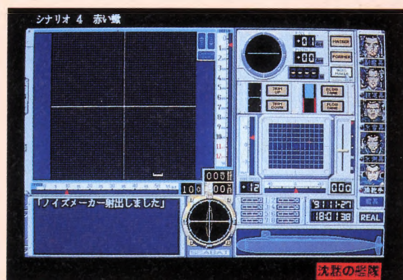
X68000 Compact対策として5インチディスクは発売されたが，高くていまだに買えない人がほとんどだと思う。「オーバーテイク」や「ふしぎの海のナディア」などの対応ソフトはチラホラと出ているものの，やはり買うものに困ってしまう（多すぎてじゃなくて，少なくて）ときもあるだろう。

そこで，見落としがちなところに目を向けてみよう。つまり，ブラザー工業のTAKERUである。TAKERUでは新作もどんどん発売しているが，昔のパッケージ売りソフトをTAKERUで復活させるということもやっている。

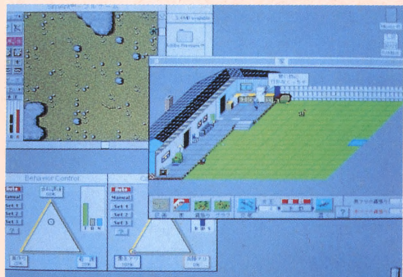
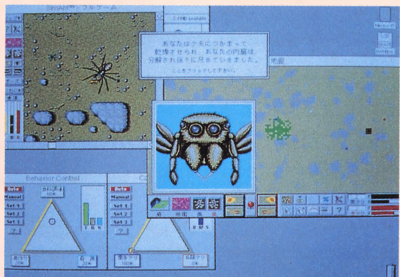
エグザクトの「ナイアス」「アクアレス」，「A III」「栄冠は君に」を含むアートディンク全製品，そして，アルシスソフトの「スタークルーザー」「ナイトアームス」などなど。値段も安くなっているし，いまだに名作と呼ばれるソフトも多い。

さらにタイトルは増えるかもしれないので，TAKERUのリストは要注目。特に「スタークルーザー」なんかは5インチユーザーでも買っていない人がいたら，超オススメのソフトだ。

沈黙の艦隊

発売が遅れているこのゲームだが，出ることは確実なようだ。グラフィックは基本的にはPC-9801からそのまま持ってくるということだが，もともとの画面が結構きれいだったので，そんなに気になるほどではない。ついでにいうと，同じジー・エー・エムの「バトル」は残念ながら発売中止となってしまった。

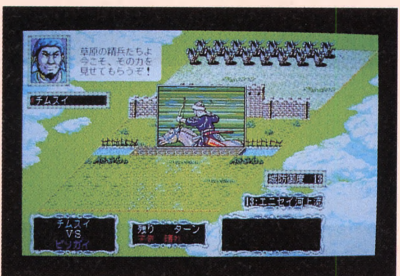
X68000用　3.5/5"2HD版　12,800円(税別)
ジー・エー・エム　☎03(3736)6879

FM TOWNSの画面です

シムアント

イマジニアが次に出すソフトは，この「シムアント」で2月下旬発売の予定。「シムアント」は“シム”シリーズの第3作目で最新作ではないが，いちばんの変わり種といえるだろう。飼育ケースでアリを育てる楽しみを土台に，“野生”のアリを育てるという，コンピュータゲームならではの内容となっている。

X68000用　5"2HD版　価格未定
イマジニア　☎03(3343)8911

（画面はMacintosh版です）

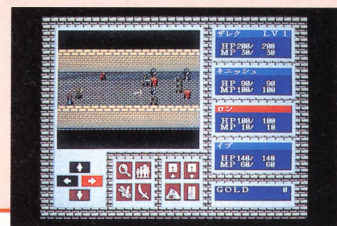
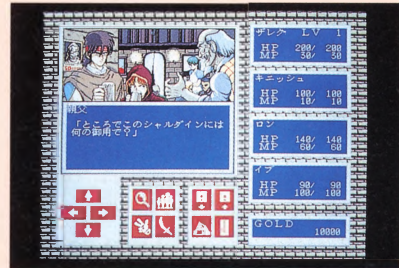
蒼き狼と白き牝鹿・元朝秘史

数年間の時間を経て，光栄の名作「蒼き狼と白き牝鹿」が蘇った。今回も“モンゴル編”でモンゴルを統一し，“世界編”へと進んでいくという内容と，ゲーム全体の流れはほぼ同じだ。もちろん，まったく同じなわけはない。時代のニーズに応えるべく，戦闘などのシステムはかなりパワーアップしている。あのオルドも健在なので，お好きな人はご安心を。

X68000用　3.5/5"2HD版　9,800円(税別)
光栄　☎045(561)6861

（画面はPC-9801版です）

ヴェルスナーグ戦乱

いろいろと新機軸が盛り込まれているという，ファミリーソフトのロールプレイングゲーム「ヴェルスナーグ戦乱」。もう1年ぐらい発売延期になっていたような気がするが，今度こそ本当に“もうすぐ発売”ということだ。崩壊したあと魔法を失ったファンタジー世界を舞台に，勇者たちの戦いが繰り広げられる。

X68000用　3.5/5"2HD版7枚組　9,800円(税別)
ファミリーソフト　☎03(3924)5727

スクエア・リゾート ハイパー戦車戦

グラマーな幼児体型の女の子が描かれているパッケージで，一見すると違うジャンルのゲームと間違いそうなソフト。しかし，内容はれっきとした（？）タンクバトルゲーム。クォータービューで表示される戦場は高低がつけられており，高いところから低いところへ流れる弾を撃ち合う，なかなかユニークなゲームである。

X68000用　5"2HD版2枚組　4,500円(税別)
ファミリーソフト　☎03(3924)5727

TREND ANALYSIS

1992年11月の月間売り上げベスト10

POINT	タイトル	発売元	発売日
1416	オーバーテイク	ズーム	'92/11/20
885	テラクレスタ/ムーンクレスタ	電波新聞社	'92/11/20
797	ストライダー飛竜	カプコン	'92/11/27
404	ロードス島戦記II	ハミングバード	'92/11/20
265	デスブレイド	SPS	'92/10/30
240	バーンウェルト	グローディア	'92/10/30
164	スーパーD.P.S.	アリスソフト	'92/10/15
126	ふしぎの海のナディア	ガイナックス	'92/10/30
139	スクエア・リゾート	ファミリーソフト	'92/11/20
55	MATIER	サンワード	'92/10/9

久びさに活気溢れるラインアップとなったと感じる11月の売り上げベスト10。

まずは予想どおり，「オーバーテイク」がぶっちぎりのトップに立った。前評判からしてすごいものであったが，それに劣らないだけの売り上げを見せてくれたようだ。ポイント数は月によって重みが変動するため，単純に比較することはできないが，過去最高に近い売り上げであることは間違いないだろう。

ズームのメーカー自身の人気，それに加えて，F1レースゲームというプラスの要素，そして，出来のいい店頭デモとくれば，これはもう完璧な布陣である。

また，この「オーバーテイク」に引っ張られて，同時期発売のソフトもいい結果を出している。もちろん，内容的にもいいものが揃っていたからであることはいうまでもない。

2位の「テラクレスタ/ムーンクレスタ」はかなり古いアーケードゲームからの移植だが，ゲームファンの心をくすぐるうまい選択である。特に「ムーンクレスタ」は，誰しもノスタルジアを感じるソフトではなかろうか。

3位は「ストライダー飛竜」。カプコンの第2作はコレということで，くだんのアレを期待した人は肩すかしをくらったかもしれない。しかし，このゲームも随所に趣向の凝らされた名作ゲームである。並のアクションゲームでは決してない。

メディアミックスものとして固定ファンも多い「ロードス島戦記II」は，4位という堅実な順位についた。前作の「ロードス島戦記」からはかなり時間がたっているが，PC-9801版「ロードス島戦記II」の発売や，ほかのメディアでの好調な動きに支えられ，人気を持続していたようだ。内容もその人気に見事応え，しっかりした移植になっている。

5，6，8位には，10月30日に発売された「デスブレイド」「バーンウェルト」「ふしぎの海のナディア」が，前回に引き続きトップ10に入っている。「デスブレイド」と「ふしぎの海のナディア」は若干ポイントを落としているが，「バーンウェルト」はその逆。順位を落としながらも，獲得ポイント数は上げている。雑誌や身の周りの評判をじっくりと見定めてから，買いにいった人が多かったのだろうか。

9位の「スクエア・リゾート」は“ハイパー戦車戦”というサブタイトルどおり，戦車どうしがユニークなルールに乗っ取って攻撃しあうアクション（パズル？）ゲーム。パッケージが女の子の絵なので，店によってはアダルトゲームの棚に置かれているかもしれない。買いにいって見つからないときは，そっちもよく探してみよう。

というわけで，前回に引き続き今回も動きのあるベスト10で興味深い集計となった。12月は新作ソフトが少なかったので，次回の集計は少し気掛かりである。

［データ集計協力店］（順不同）
九十九電機本店
J&P（渋谷/町田）
OAシステムプラザ横浜店
P&A
ラオックスGAME館

ウワサのソフトウェア（海外編）

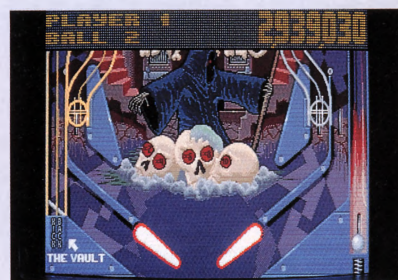
PINBALL FANTASY

ゲームセンターにあるゲームはほとんど日本人が作ったゲームであるが，ピンボール（フリッパー）だけは日本製のものを見かけない。気のきいたゲームセンターに行けば，コンピュータ制御の電飾もまぶしい，西洋生まれのピンボールたちに会えるだろう。

彼らは向こうの娯楽文化でも，かなり中心的部分を占めている。ジュークボックスと並んで，映画やポスターといった分野での彼らの役割は，言葉に表しがたい。身も心も日本人のわれわれは，そんな憧れをちょっとポケットに入れて，両手を掛けるのである。

ピンボールをコンピュータ上に再現することは，昔から数限りなく行われてきた。現実の質感こそやや失われるものの，ピンボールの台が手軽に自分のものになることには，代えがたい何かがあるのだろう。

このゲームから感じられるのは「カエサルのものはカエサルの元へ」という言葉のとおり，ピンボールのコンピュータ化も西洋人がやることで，１枚も２枚も上手のものができるということである。

プレイ画面は縦に３画面ほどあり，ボールを

追って画面をスクロールさせることで，縦長の盤面を再現している。これは前作の「PINBALL DREAM」と共通のシステムになっているが，馴れるまでは見えない部分を把握できなくてつらい。しかし，結局はそれほど遊びにくいとは感じなくなるのには，大いに感心できる。

気になる台は，全部で４種類から選択可能になっている。簡単にいってしまえば異なる４つのピンボールが遊べてしまうわけだが，そこはそれ，ボーナスのメカニズムなどに，ある程度の共通点を持たせることで，どの盤にも不思議な遊びやすさを感じることができる。

モチーフにされているデザインも，ピエロや車に，億万長者やホラーコミック，といった定番のものをもらさず踏襲しており，アーケード気分を盛り上げてくれる。普段から玉を弾いていて詳しい人には，類似台を推理する楽しみもあるかもしれない。（八）

発売元　21st Century Entertainment

ウワサのソフトウェア（海外編）

ROAD RASH

レースゲームというと，たいていはライバルの車などをうまく避けながら，タイムを競い合うものである。普通のアクションゲームとは異なり，敵を攻撃したりなんてことはできないし，だいたいチンタラそんなことを考えていたら，ほかの車に抜かれたり，コースアウトしたりして順位を落とすのが関の山だ。

しかし，世間は広いと相場は決まっているので，そうでないものもたまにある。身近なところでは，敵を追いかけ攻撃する「チェイスH.Q.」がそれにあてはまる。あのゲームでは指定された車に攻撃をして停車させるのが目的だったが，ほかの車にぶつかったりしてもスピードが落ちるだけだった。

この「ROAD RASH」は，ライバルを殴ったり蹴ったりしながらゴールを目指す，オートバイのレーシングゲームだ。もちろんライバルたち

もこちらに攻撃してくる。ときには２人がかりではさみうちにされ，袋叩きをくらってしまうこともあるから油断できない。また，棒切れを拾えば武器になる。

アーケードゲームにも，オフロードバイクで似たようなことをするものがあるので，発想自体は特に目新しいというわけではないようだ。実は世界のどこかにこんなレースが実在したり

して……。でも，ストックカーレースみたいに車をぶつけ合うのはまあ納得がいくとして，ドライバーがお互いに殴り合うのは……，やっぱりなさそう。

で，こういうルールもユニークながら，実は最高に面白いのが，殴られたり，障害物に当たったりして，落車したとき。バイクを取りに走り回らなければならなくなるのである。

ちょこまかと走るドライバーが，レバーの操作どおりに前後左右に動き回る。この間に，バイクがほかの車にはねられて遠くに行ってしまうこともある。ピョーンと飛んでいくバイクを見るのはなかなか悲しいものがある。やっと追いついてバイクにまたがろうとした瞬間に，後ろから来たバイクにひかれたりなんてときもあり，実は落車してからが真のゲームの姿なのでは，と感じさせる。いつも思うけど，外人ってヘンなこと考えるのが得意だな。

発売元　ELECTRONIC ARTS

待望の最強将棋ソフト登場

Yamato Satoshi
大和　哲

将棋ソフトは強さが命。機能がどんなに高くても，思考ルーチンがへボければしょうがない。というわけで，徹底的に強さを求めたと銘打たれた将棋ソフトがここに登場した。いうだけのことはあって，本当にかなり強いみたいだぞ。

毎年冬の恒例行事となりつつある第3回コンピュータ将棋選手権が，昨年の12月6日に東京千駄ヶ谷の将棋会館で行われた。このコンピュータ将棋選手権はコンピュータ上の将棋ゲームどうしを対戦させて，最も優秀な思考ルーチンを持つ将棋ソフトを決定する，という大会である。

一昨年12月の第2回CSA選手権において，初参加でありながら「森田将棋III強化版」について2位の座に輝いたのが，ログの「極」である。その「極」が今年は激戦の末に優勝。柿木将棋と森田将棋に引き分けた以外は全勝で，5勝2引き分けという好成績を残した。

この「極」というソフトはPC-9801用に開発され，大会では486/66MHzのマシンで動いていたのだが，今回X68000に移植され，発売されることになった。

今回はサンプル版が編集室に届いたので，日本一の将棋ソフトの秘密と実力を見極めてしまおうというわけである。さて，どんな秘密が隠されているのか？

詰め将棋を解かせる

さて，例によって例のごとく，恒例の詰め将棋を「極」に解かせてみる。

将棋ソフトといえば，評価は当然思考ルーチンの優劣によってほとんどが決まるわけだが，少なくとも詰め将棋ができなければ，同じような局面が展開される指し将棋の終盤戦でも苦戦することが目に見えている。いわば詰め将棋は思考ルーチンの試金石といってもよいわけである。

問題は1992年7月号で「棋太平」と「将棋聖天」に解かせたもので，マシンも同じ10MHzのX68000である。結果は表1のとおりだ。参考のために前回詰め将棋で優秀な成績だった「棋太平」の記録も参考のために載せておいた。

やはりというか当然というか，「極」もすべて規定どおりの手数で解くことができたが，ここで注目してほしいのは「棋太平」との思考時間の差である。

3手詰では「棋太平」が勝ったものの，7手詰ではすでに逆転して「極」のほうが先に解き終わり，9手詰で倍近くの差，13手詰Bでは5倍もの大差がついた。

計算自体は単純だが，その量こそが問題になる詰め将棋の思考ルーチンでは，5倍の差というのは特にコーディングの際の差とは思えない。一般に詰め将棋の思考ルーチンはαβ枝刈りなどを行って，無駄な手を考えないようにしているのだが，「極」では，思考中に無駄な手を考えるのをいっそう省くような思考アルゴリズムの開発に成功しているようだ。

最近の詰め将棋の解法プログラムでは，ゲーム木をかたっぱしから解いていくのではなく，あきらかな悪手を読まずに飛ばしてしまう，いわゆる前向き枝刈りによって思考時間を減らすのがトレンドであるようなのだが，そのような方法を使っているのではないかと思えるような速さだ。

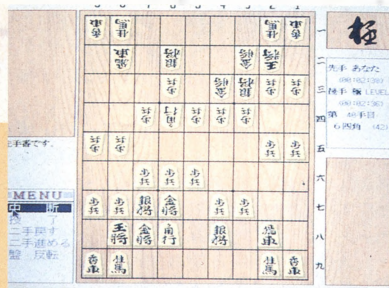
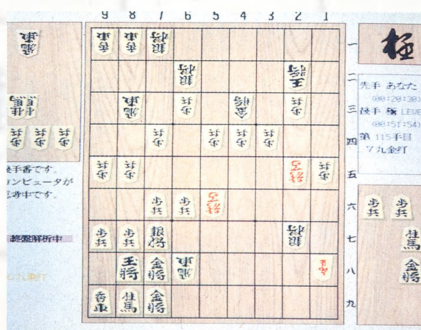
ところで，負けてしまった「棋太平」であるが，これも悪いプログラムであるというわけではない。いや，むしろ，可能な指し手の少なかった3手詰では勝っているのだから，むしろコーディングテクニックでは「棋太平」のほうが勝っていた可能性もある。ただ，詰め将棋にしろ，対戦将棋にせよ，思考プログラムというのはコーディングテクニックよりなにより，アルゴリズムこそがものをいう世界なのだ。「棋太平」にはぜひともアルゴリズムを練りこんで再挑戦をしてほしいものである。

人間との指し将棋では

さて，続いてはお待ちかねの指し将棋のほうである。今回は残念ながら時間の都合で，コンピュータどうしの対戦は行わなかった。まあ，強いことはわかりきっているので，どんな手を打つのかを分析しようと，私が「極」と対戦してみた。

いや，強い。とにかく強い。

正直なところ，私はお世辞にも強いといえるような腕ではないのだが，それでもいままでの将棋ソフトと一線を画すのがわかるくらい強い。というか，なんでこんなにいい手が指せるのか不気味ですらある。

この「極」では，コンピュータの思考レベルは大きく分けて，ノーマル思考とハイパーモード，そのなかでいくつかのレベル

X68000用　5"2HD版　12,800円(税別)
ログ　☎03(3837)2595

思考中に何を考えているのかが表示できる

詰め将棋を解くのも得意

があるため，全部で10段階の強さがある（ただしこれはサンプル版であるので製品版では変更の可能性もあるそうだ）。

まず，比較的高いレベルで考えさせると，人間から見てあきらかに悪手とわかる手を指すことがほとんどない。一般的なコンピュータ上の将棋ソフトではどんなにレベルを高くしても，コンピュータ将棋にありがちな，人間から見るとひと目でわかる悪手を指すことがあったのだが，この「極」にはそれがめったにないのである。

棋譜データの読み込みももちろんできる

残念ながらレベル０に負けてしまった

また，何度かこちらが矢倉囲いを作ったとき，いつのまにか相矢倉になってしまっていたことすらあった。いい手，いい手を指すうちに偶然そうなってしまったのだろうが（というのもまだ，コンピュータ側が美濃や船囲いするところを見たことがないからだ。偶然でない可能性もあるが），それにしても不気味である。

また，スピードの面でも格段に速い印象を受ける。ベンチマークを行ったわけではないが（だいたい強さ自体が厳密にはわからないのでなんともいえないのだが），ほかの将棋ソフトと同じ強さのレベルで比べると，思考時間はかなり短いと思われる。

この２つから推測すると，このプログラムはかなり変わった評価関数を使い，前向き枝刈りなどを行っているのではないだろうか。極力無駄な手を考えないようにして１手あたりの思考時間を短縮し，なにかヒューリスティックな情報によって，重要そうな手の候補だけについて考えるようにしているのではないのだろうか。

このソフトでは「思考表示あり」モードといって，どの手を打とうか考えているのを見ることができるように設定できる。それで見ていると，まず，挙がっている手の候補がかなり少ないのだ。

ただ，これだけだと説明のつかない現象も起こる。たとえば思考レベルを最高にしていても，最初の何手かは人間が指すのと同じようにさっさと指してくる。そして，どちらかがとれるような駒が存在すると，今度は候補の１つひとつを何度も何度も繰り返して考えるようになるのである。

表１

	極	棋太平（参考）
3手	0：02	0：01
7手	0：28	0：33
9手	0：30	0：55
13手A	0：14	0：36
13手B	2：09	11：14

最初は静かな局面のときとそうでないときでは評価関数を変えているのかとも思ったのだが，それにしては静かな局面でなくても長考しないときもある。なんらかの定跡データを持っているというようなことも考えられるが，正直なところよくわからない。

ま，ともかくも，この「極」は間違いなく最強の将棋ソフトだ。いままでの将棋ソフトと結局は変わらないなどと思っていたら痛い目にあう。人間と対戦させても，中級者あたりとならおそらく遜色ないほどの実力であることは間違いない。

そのほかの装備について

さて，最後にこのゲームの装備や気づいた点について述べる。操作法はフルマウスオペレーション。操作感覚も「棋太平」や「将棋聖天」などとほぼ同じだ。

また，盤面設定は自由で，盤の反転もワンタッチでできるが，駒落ち設定は残念ながら飛車，角２枚落ちだけである。

「棋太平」や「将棋聖天」にあった読み上げ機能はないが，将棋ゲームの本質である思考性能や操作性に関係のある部分ではないのでいいだろう。むしろそのほうが，メモリを大量に使わざるをえない将棋ソフトではいいのかもしれない。

思考時間についてだが，低いレベルではかなり速いが，レベルが高くなるとやはり遅い。しかし，もともと思考時間を無視して強さを追い求めたというのだから，それなりに遅くなるのは当然だろう。先行機種とのCPUパワーの差を考えれば，むしろよく健闘しているほうだと私は思う。

とにかくこの「極」はよくできたソフトだ。しばらくの間は，X68000上でもほかのパソコン上でも，互角のものは出ても，これを簡単に凌ぐようなソフトが出ることはおそらくないだろう。

読者の諸君には，パソコン上での将棋に興味があるのならば，絶対に買ってほしい１本であるといっておこう。そして，ほかのソフトハウスに対しては，さらなる乱入を求める次第である。

マシンパワーがほしい

これでＸ68000上にも最高レベルの将棋ソフトが登場することになったわけだ。

それほど将棋の強くない私には，もうどこがどうと弱点を指摘することはほとんどできなくなってしまった。あえていうなら，終盤に少し弱さがあるような気もするのだが，高レベルになると一度も勝てない私にはあまり自信はない。これを考えると，野球の解説などは本人がたいしてうまくなくても解説できるようだから，不思議なものだ。

さて，こうなってくるとあとはスピードの問題なのだが，現状では１手に１秒～20分くらいかかる。しかし，もともとこのソフトが目指して作られたCSAの選手権では持ち時間45分，時間切れ＝即負けというルールを使って勝負をしている。それを考えると，このソフトもそこそこのCPUパワーさえあれば問題ないレベルのスピードにアルゴリズムレベルでは達しているはずで，遅いソフトということはできない。少なくとも，45分という時間は将棋の世界では決して長いという時間ではない。

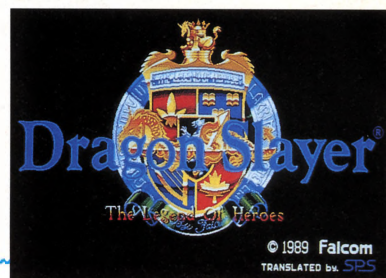
このテの思考時間がメインとなるソフトでは，どうしてもCPUパワーのある機種というのがほしいものだ。もちろん，ソフト面でのスピードアップ，思考強化が最重要なのだが，それにも限界が見え始めているから。

総合評価（0〜10）

項目	評価
攻め強さ	★★★★★★★★★
守り強さ	★★★★★★★★
詰め将棋	★★★★★★★★★
操作性	★★★★★★★

国民的RPGのおでましだい

Nishikawa Zenji
西川 善司

"ドラゴンスレイヤー"は日本ファルコムのドル箱ともいうべき人気シリーズである。そして今回，そのなかでは比較的新しい作品である「ドラゴンスレイヤー英雄伝説」が，SPSの手によって移植された。

ドラスレ大好きっ子(死語)

突然だが，私は「ザナドゥ」が好きだった。どのくらい好きかというと，「早解き全国3位の終了認定証カード」を獲得したくらい。ちなみに，この「ザナドゥ」は別名「ドラゴンスレイヤーII」という。

また，私は「ソーサリアン」も好きだった。どのくらい好きだったかというと，これをやりたいがためにPC-8801FHを買ってしまったくらい(あとでX1turbo版が出た。がちょ～ん)。ちなみに，この「ソーサリアン」は「ドラゴンスレイヤーV」ということになる。

そして今回紹介する「英雄伝説」は，ドラゴンスレイヤー（以後ドラスレと略）シリーズ第6作にあたる。思い返せばこのシリーズは，PC-8801とかX1とかの8ビットマシンが現役の時代から続いているのだな。シリーズ1作目とかは「タモリ」顔のモンスターとか出てきちゃったりして，結構おちゃらけていたっけなあ。

それが2作目の「ザナドゥ」以降は一転してシリアス路線。ファミコンやMSXシリーズなどにも「マジメ」なドラスレシリーズは浸透し，当時ドラスレシリーズが発売された機種は「安泰」というジンクスまであったくらい。

さーて，X68000にもこのドラスレシリーズが発表となったわけだけど，X68000は「安泰」か。うーむ。むひょ？

X68000用　5"2HD版　9,800円(税別)
SPS　☎0245(45)5777

やっぱり外に出させてくれない兵士たち

時代の映し鏡「ドラスレ」

ドラスレシリーズは毎回違ったタイプのゲームで登場した。今回の「英雄伝説」はいわゆるエニックスの「ドラゴンクエスト」タイプ。私の大嫌いな「戦う」「呪文」「逃げる」とかを選ばせるタイプだ。

「ドラスレシリーズは代々アクションゲームの要素が盛り込まれていたのに，いったいなんだあ？」

と叫び出したい人もいることだろう。私は叫んだ。しかし，よくよく考えてみるとドラスレシリーズは毎回ユーザーのニーズに合ったタイプで登場してきているのだった。たとえば「ザナドゥ」「ロマンシア（ドラスレIII)」「ソーサリアン」の頃はとにかくアクションRPGが流行った，売れた。

ドラスレVII「ロードモナーク」はシミュレーション（パズル？）タイプのRPGだ。これは昨年，一昨年からのパズル/シミュレーションゲームブームに便乗して出されたものに相違ない。

そして「英雄伝説」のオリジナルPC-8801版が発売されたのは，やはり例のドラクエパニックの頃に合致する。

ドラスレはまさに時代の映し鏡なわけ。だからもしかしたらさ，1対1の対戦格闘タイプのドラスレなんかが登場したりしてね。しょーりゅーけんっ！　いや～ん。

英雄伝説

前置きが長くなっちゃったけど，「英雄伝説」は，よーするにドラクエなんだな。だけどさすがファルコムだけあってまとめ方は上手。単なるクローンゲームという言葉では片づけたくないほど，スマートにまとまっている。

自分が扮するキャラクターはセリオス王子という王家の純血，16歳の金髪の美男子。決して，うどん屋とかのアルバイト店員の加藤君とかじゃないあたりがセオリーだよな。安い時給でこき使われる花粉症気味の若僧じゃなくって，毎日剣術と学問と魔法の勉強にいそしむぼっちゃんなのだ。んで，舞台は干からびた小口切りのネギが散らばる調理場じゃなくて，イセルハーサと呼ばれる剣と魔法の世界。

小さいけれど人々が幸せに暮らす王国ファーレーンは心やさしき王アスエルのもとに統治されていた。けれど突然のモンスターの襲撃によってアスエルが暗殺されて，

町の外に出るとフィールド画面に

この画面なんか，いかにもドラクエって感じ

アイテムにはいろいろと面白いものがある

世継ぎの王子は当時6歳。王子が王位継承にふさわしい年齢になるまで側近のアクダムが摂政となり，政務を引き継ぐことになる。ここまでが前置きで，物語は王子が16歳になった10年後の世界から始まる。

え？　なに？　加藤君がいうには，このアクダムっていうやつが怪しいってさ。そうだねえ，たしかに話がうまくできすぎている。モンスター襲撃事件で一番おいしい思いをしたのは，摂政となったアクダムだもんねえ。えっ，「アクダム」→「アクダマ」→「悪玉」と連想できるからだって？……はいはい。

王子の旅立ち

王子セリオスは，王位継承のその日まで，城から離れたエルアスタという町で爺やと暮らすことになった。ゲームは，王位継承の儀式まであと2カ月というある朝から始まる。

召し使いのお姉さんが起こしにきてくれた。はー，いい朝だ，といっている間もなく爺やが現れて，イッツァ勉強タイムだぞうな。爺やの隙をうかがって，屋敷の中を歩き回ろう。夕食は王子さまの大好きなビーフシチューだそうだ。屋敷が飽きたら今度は町に出よう。

ゲーム全編を通してのキー操作はシンプルだ。テンキーとリターンキーorスペース（決定）とESCキー（取り消し）だけ。ジョイスティックももちろん使える。ジョイパッドなんかで遊ぶとファミコン気分だね。

町の人と話すのも飽きたので，町の外に遊びにいきたいところだが，門兵は爺やのいいつけとやらを守って外に出してくれない。そこで近くの畑をいじってる農夫に話しかけてみると，「えへへ，王子さま。いい娘がいまっせ，こっちでげす」と怪しげな店に案内してくれるわけもなく，町の壁に秘密の抜け穴を作ってくれていて，これを使って草原へと出られた。

町の外へ出ると，イセルハーサ全体マップの移動モードになり，町中での移動モードと同じ感覚で移動が可能。町では町の住民やそのほかの人が動いているのが目に見えていたよね。だから，町の人と話したいときはそのキャラクターの側にマイキャラを持っていけば話せたわけだけれど，全体マップモードでは画面に映っているのは地形とマイキャラだけ。

戦闘画面で敵を倒すと上に飛んでいく

このマップモードで歩き回っているとモンスターに遭遇しちゃったりするけど，モンスターは目に見えない。不本意な戦闘を強いられることもあるけれど，まあ，これはこのゲームのゲーム性だと思うか，「ドラクエがそうだから，そういうものなんじゃないの？」というふうに思い込むかして納得するように。ちなみに，このマップモードで徘徊するモンスターを透視する魔法アイテム「あらわしの鈴」というのがいちおうあるけれど，ただじゃ手に入らないからねぇ。

敵は遭遇して初めて画面に現れる

さて，草原でスライムをいじめるのに飽きて町に帰ってきてみれば爺やがカンカンに怒って待っていた。そりゃそうだ，お勉強中に抜け出したんだもの。長いお説教のあと床につくセリオス王子だけど，その深夜，恐ろしい事件が巻き起こって……。

ゲームシステム

ゲームは全部で6章構成。それぞれの章につき，大きなクエストが1つあって，そのクエストが終了すると，その章のクリアとなる。もちろん全章にわたって張られた伏線や謎は，すべての章をクリアしなければあきらかにされない。この「英雄伝説」がどうして「**ドラゴンスレイヤーⅥ**」なのかも，終章になって初めてうなずける。

このテのゲームは「シナリオ命」であり，必要以上にあらすじを書くとシラケてしまうので，これ以上バラさないことにする。んでは，私がプレイした感想でも述べさせてもらおう。

戦闘時のメッセージもドラクエっぽい

こういったシチュエーションもある

シナリオはありがちで先読みできちゃうときもあったけど，そこはそれ，「遠山の金さん」や「水戸黄門」と同じでマンネリならではのよさと爽快感があった。

各キャラクターの性格づけも時代劇みたいに明確で，つい感情移入してしまう場面があったりした。ただ，キャラクターの顔や姿のグラフィックっていうのはゲーム中あまり出てこないのが残念。あと，前半はヒロインが出てこない（というか誰だかわからない）男臭い物語になっているけれど後半には次第に物語の要になってきて，むふふ，ラブトライアングルができちゃったりして，セリオスのやつ結構憎いのねん。

あとはゲームデータのセーブにその時点のキャラクターのレベルやセーブした場所がメモされるのは最高に便利だった。また，メッセージや経験値の表示のコンフィギュレーションができちゃったりするのもなかなかいいぞ。

ゲームに慣れてくるとサクサクとシナリオが進んでちょっと紙芝居みたいになってしまっているところもあったけど，最後まで解き終えたときには安堵感と感動が押し寄せてきて，つい「はふっ。ちょっとため息」なんて死語が飛び出しちゃったよ。いや，ほんと。

オートマチック戦闘モード

んで，なんで私がこのテのRPGが嫌いかっていうとね，戦闘モードがたるいからなの。メニューから「戦う」とか選んだりするのって，なんかゲームしてる気がしないでしょ。ツール使って仕事してる気になるよ。だいたい，このテの戦闘モードってモンスターと遭遇したっていう緊張感の演出に見事に失敗していると思わない？

―閑話休題―

「英雄伝説」の戦闘モードには最近のRPGによく採用されている「自動戦闘モード」も搭載。このモードを設定しておくと，戦闘時のうざったい「戦う」だとか「呪文」だとかのメニュー選択を自動的にやってくれる。しかし，これが頭悪い。このモードは体力の低いモンスターをみんなで集中攻撃する，という単純なアルゴリズムのもとで動いているみたい。

攻撃力が強い戦士は体力の高いモンスターの相手をして，魔法使いは戦士たちの援護をするとか，魔法を使う小癪なモンスターから分担して倒していく，とかいった気のきいた行動はとってくれない。リンチみたいに「よーし，こいつシメちまおうぜ」「よしきたっ」「ボコボコボコ」「……ふぅ，んじゃ次こいつな」「よしきたっ」「ボコボコボコ」みたいなチンピラ的思考の自動戦闘モードに成り下がっている。

このへんは今後の研究課題だろうね。自動戦闘モードのアルゴリズムを数種類から選択できてそれを各キャラクターごとバラバラに設定できると，いまよりずっとマシになるだろうね。

それにしてもさ，こういった戦闘モードをわざわざ自動化するような動きが浸透してきているっていうことはさ，それだけ戦闘がうざったいっていうことをゲーム会社自身も認めているわけだよね。なんかパラドックスだねぇ。

終わりに

なんだかんだいっても，遊び出すと最後まで遊びたくなるゲームだ。はっきりいってハマリがないゲームなので難易度は低い。だから，ふだんはこういうゲームをしない人にもオススメだね。子供たちが夢中になっている「ろーるぷれいんぐげーむ」っていうのはいったいどんなものなのかな，という子供を理解しちゃろうっていう向学心旺盛なお父さんお母さん，学校の先生にもお勧めしちゃうね。あ，もちろんうどん屋の店員の加藤君にも，ね。

うーん。しかし，最後はなんとなくエコロジーなテーマだなあ（あっ，いっちゃった）。

ドラスレシリーズよ，永遠に

んじゃ，外周りのお話を少し。

BGMはAD PCM未使用，FM音源だけで奏でられていて，ほかの最近のゲームと比較すると地味な印象。だけどゲーム画面にマッチしたものがたくさん用意されているので，ゲームの雰囲気は盛り上げてくれる。戦闘のテーマは死ぬほど聴かされるけどイヤにならないのは，もしかしたら名曲だからか？

グラフィックはPC-9801版そっくり。ドット比の関係でちょっと横伸びしている感じがするけど，そんなには気にならない。戦闘モードでのモンスターの絵はキレイだけれど，静止画なので「ドラゴンのファイヤーブレス攻撃をくらった」とかいわれても，ドラゴンちゃんは遠くを見つめたまま動いてくれない。なんか拍子抜け。せめて，攻撃したときくらいはアニメーションすればよかったのにね。

さてさて，PC-9801版では「英雄伝説II」も発売されたみたい。X68000版の「I」を出してドラスレ気分を盛り上げさせた以上は，責任取って「II」も移植してね，SPSさん。

総合評価	0　　　5　　　10
シナリオ	★★★★★★★★★
サウンド	★★★★★★
グラフィック	★★★★★★
システム操作性	★★★★★★★★
ゲームスピード	★★★★★★★★
登場人物	★★★★★★★★
戦闘モード	★★★★

美少女とメカとアニメ

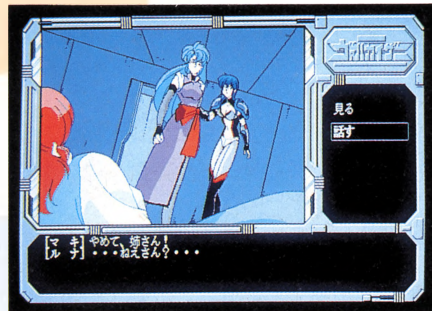
Takahashi Tetushi
高橋 哲史

一部にウケそうな要素をとりあえず集めて，アドベンチャーにした，という感じのゲームがこの「機甲装神ヴァルカイザー」だ。いかにもという女の子といかにもというメカが，いかにもというアニメーションをする。

事件は唐突に

岬博士「な……，なんだ君たちは」
謎の男「岬博士，あなたの開発したバイオローダーを我々に渡していただきたい」
岬博士「バイオローダーを？　断る。あれは使い方によっては悪魔にもなる。もう２度と人目にさらすこともない」
謎の男「ふっ。そういうだろうとは思ってたがな……」
レイカ「博士！」
岬博士「レイカ君！」
謎の男「素直に渡せばよし，さもなくば助手の女の命も保証できなくなるが」
レイカ「博士，お願いです。研究を渡してください。そうしないと，私ばかりか博士や，ルナちゃんまで」
岬博士「くぅ……」
謎の男「どうするね？　博士」
岬博士「レイカくん，すまない。この研究は何があろうと渡すわけにはいかない」
謎の男「けっ，強情なやつだ。しかたない，ボスのいいつけどおり，全員仲良く死んでもらうとするか」

翌日謎の爆発により崩壊した研究所の焼け跡から，奇跡的に岬博士の妹，留奈（ルナ）が無傷で救出された。しかし，すべての研究は失われ，岬博士の消息もいっさい途絶えてしまったのであった。

戦闘ロボットなどのメカもウリのひとつ

このテの女の子が多数登場する

ルナちゃん，ファイトっ！

といきなり燃える展開で始まってしまうこのゲームですが，中身は基本的にコマンド選択式のアドベンチャーでゲームのほうは案外サクサクと進んでいくのでした。プレイヤーは予想どおり岬留奈ちゃんになるわけで，しかも序盤では舞台に女子高が登場したりして，その筋の人にはたまらないでしょう（おいおい）。

研究所の謎の爆発から一夜明けた朝。たったひとりの肉親である兄を失った悲しみがベッドの中のルナをおおう。しかしルナは希望を捨てない。

こらえていた涙がまた溢れそうになったとき，後ろからルナを呼ぶ声がした。振り向くと家政婦ロボットのVALがいる。

さびしいながらもいつもどおりVALと２人の朝食をすませると，級友のユミが迎えにきてくれた。

ユミ「ルナ，おはよーっ！」
ルナ「おはようユミ。じゃ，VAL，行ってくるわね」
VAL「行ってらっしゃい」

元気を取り戻し，学校へ向かうルナ。しかしその笑顔に第２の陰謀がふりかかりつつあることを彼女はまだ知らなかった。

こ，ここはまんがの……

スッスッスと

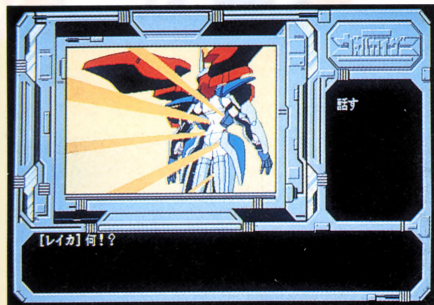
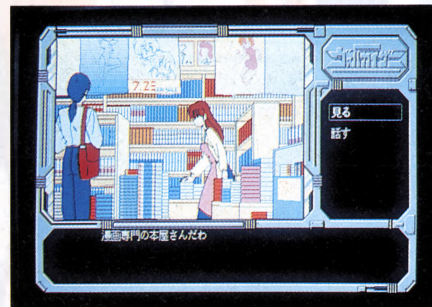
結論からいってしまうと，全体的にややものたりない感じです。なにしろ肝心のゲーム自体がほとんど詰まることなく１本道で進んでしまい（１カ所だけ迷うことは迷いましたが……），３時間ほどでエンディングまでたどりつけてしまうのです。まあ，お手軽なアドベンチャーが好きな人にはいいかもしれません。

絵や音は残念ながらPC-9801からのベタ移植ですが，それなりには充実しています。止め絵部分ではちょっと厳しいかな，と思う絵もありますが，ウリであるアニメの部分はきっちりと動かしています。グラフィック担当の人はもしかしてアニメータさんなのかな。

“兄の仇を討つために，謎の兵器組織に敢然と立ち向かう美少女”なんて設定が好きな人にはいいでしょう。私はとりあえず長々と予告が入っていた次作に期待することにします。頑張ってください。

あのあたりではウケるに違いない

美少女にメカ。そしていかにもアニメな絵柄（どことなく“るーみっくキャラ”っぽいと思うのは私だけでしょうか？）が揃っていて，狙いを定めている層がありありとわかります。ゲーム中にはまん○の森とかも出てくるしね。

総合評価	0 ― 5 ― 10
シナリオ	★★★★
操作性	★★★★★
ＢＧＭ	★★★★★★
アニメ	★★★★★★★
お手軽度	★★★★★★★★

X68000用　3.5/5″2HD版　4,800円（税別）
ブラザー工業（TAKERU）☎052(824)2493

オリジナリティあふれる知性派RPG

Shibata Atushi
柴田　淳

“ダンジョンに侵入してくる勇者たちをやっつけろ”。こういうと「ウィザードリィⅣ」や，少し前の“THE USER'S WORKS”に載った「Sim Dungeon」みたいだけど，この「キングス・ダンジョン」もそういうゲームなのである。

ファンタジーといっても，どこが面白いのかピンとこない時期があった。設定がアマいというのか，いくらおとぎ話にしても，ある程度論理的な組み立てで読む側を納得させてくれないと，物語に集中できないのだ。コボルトとかオークとかいうのは，いったいなんで人間を襲うのかとか，魔法を実現するためのエネルギーはどこから得るのかとか，本を読んでいるとついつい考えてしまうのだが，本の中ではそれらに関して触れられていない。

そんな疑問にひとつの答えを与えてくれたのが，ラリー・ニーヴンの「魔法の国が消えていく」という1冊の本だった。ニーヴンはノウン・スペースと名づけられた宇宙史シリーズなどで知られるSF作家である。彼はその本の中で，マナという一種のエネルギーの概念を導入したのである。そのマナというのは魔法の源であり，モンスターたちの遺伝的不安定性を補償しているというのだ。物語は長い年月を経て，マナがなくなりかけた世界を舞台に展開する。

ビッグバンから世界の終末まで，エントロピーはひたすら増大し続ける。そこを踏まえつつ，いかに限界を打ち破るかということにSF（特にハードな）の面白さが潜んでいると思うのだが，ニーヴンがそんなSF的な設定を持ち込んでくれたおかげで，僕にとってファンタジーはぐっと読みやすいものとなった。魔法力はまったく無尽蔵ではないし，モンスターたちもいずれは滅びゆく運命をもっていたのである。

RPGを作るときも，そこらへんのことを頭に入れておかないと，ひどいものが出来上がる。最低限，「エントロピーは常に増大」という大原則を感じさせてくれないと，ゲーム世界に共感できない。やたらとハデな魔法を連発したり，ファンタジーという設定だけに酔ってしまって，どこかで見たようなストーリー展開だったりするのは，ちょっとゴメンである。

で，この「キングス・ダンジョン」というゲームはパッと見は地味だけど，思い入ればっかり強くて内容が空回りしているようなRPGなんかよりは，ずっとマシなゲームである。第一オリジナリティがあるし，独自の雰囲気を作り出すことに成功しているのだ。

魔王になる快感

コペルニクス的転回というのだろうか，このゲームの主人公は勇士でもなければ魔法使いでもない。ダンジョンの最深部にいて，英雄を倒すべく手ぐすね引いている魔王なのだ。プレイヤー自らが魔王となり，冒険者たちの侵入を阻止するのがこのゲームの目的である。「ウィザードリィIV」なんかと同じだ。

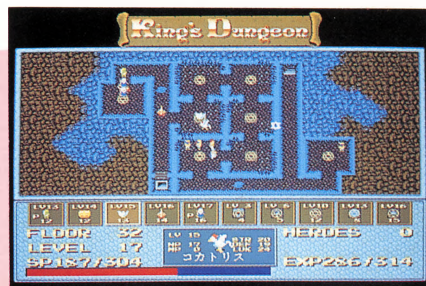
50の階層に分けられているダンジョン。それぞれの階は細長い道と，道につながれたいくつかの部屋からなっている。各部屋の中心には魔方陣が描かれており，プレイヤーはマウスを使って，そこに魔王の刺客であるモンスターや各種の罠を配置できる。が，無節操に配置できるかというとそうではなく，そこはスピリット（精神力ということだろうか）というパラメータがちゃんと用意されていて，モンスターを生み出したりするごとにその値が減っていく。ちょうどシミュレーションでいうユニットの生産みたいなものだ。強いモンスターを生み出すためには多くのスピリットを必要とする，というのはお察しのとおりである。

各階に現れる勇者たちが勝手に動き回るのはもちろんだが，魔王の生み出したモンスターもまた，彼らの行動パターンに乗っ取って勝手に動き回る。動き回るうち，お互いハチ合わせると戦闘が始まる。戦闘に関しても，プレイヤーはいっさい手出しはできない。だから，プレイヤーはまず第一にモンスターの行動パターンを熟知し，うまい位置に呼び出さないと，敵に遭遇しないまま彼らをうろつかせることになる。

さて，その敵どもは放っておくとどうなるのか。勝手に動き回って，ダンジョンの中の宝箱を見つけたりする。中には剣とか防具が入っており，彼らはどんどん強くなる。しまいには，下の階に続く階段を見つけ出して，まんまと抜け出してしまう。

話は少々前後するが，モンスターを生産して減ったスピリットは基本的に敵を倒すことによって増やすことができる。それと同時に，俗にいう経験値ももらえるのであ

X68000用　5"2HD版　5,800円(税別)
ソフトプラン　☎08669(3)8686

最初のうちは弱いキャラで我慢の戦い

キャラをクリックしてステータス表示

る。経験値が増えていくと，段階的により強いモンスターを生み出せるようになる。

では，数人の勇者が階下に逃れた場合はどうだろうか。当然経験値がもらえないから，その階で敵を全滅させれば召喚できるようになったはずのモンスターを生み出せない状態のまま，次の階へと進むことになる。逃れた敵というのは直下の階に現れるので，当然その階の難易度が増す。

痛恨の一撃とは，ありがちなセリフ

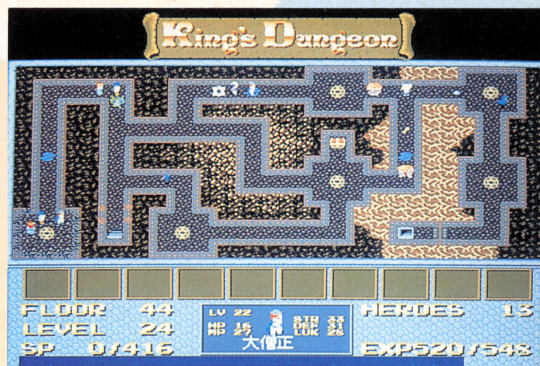
だんだんと強いキャラが登場してくる

モンスターはいつも期待どおりに動いてくれるとは限らず，せっかく召喚したモンスターも勇者に出会わないまま，というようなハプニングはしょっちゅうである。

パーティ戦の魅力

要するに僕はこういいたいのだ。このゲームは決して複雑なルールをもっているわけではない。しかし，それぞれのルールが有機的に結びついていて，ゲームに深みをもたせている。同じマップで同じような登場人物が出てくるのであっても，ゲームを進めるとそこにバリエーションが生まれ，毎回違ったシチュエーションでプレイヤーを楽しませてくれる。

そのバリエーションにさらなる広がりを付加しているのがパーティ戦というシステムである。魔王の敵となる冒険者たちがパーティを組む，ということはたやすく想像できるだろう。彼らは仲間を見つけると，そのあとに続きパーティを形成するのだが，それとまったく同様なことが，魔王の生み出すモンスターたちにも起こるのだ。基本的にモンスターには徒党を組む性質の種類とそうでないものがあり，前者はお互い連なってダンジョン内を歩き回る。

モンスターや数々現れる勇者たちの能力には当然個性があり，接近戦に長けたものや，魔法が使えるものなどさまざま。で，腕力はないが飛び道具が使えるキャラクターは後列に配置する。これがパーティ戦の常識である。やはり魔王様としては，勝手に動き回るモンスターたちを，そのような効率のいいパーティに仕立てたい。

操作に慣れてくると，実際にそういうことができるようになる。そればかりか，敵のパーティの列の真ん中あたりにいる腕力の弱い奴を横っちょから攻撃する，なんてこともできる。

このゲームでは，常に経済性を考える，つまりいかにスピリットをケチるかということが勝利のカギを握っているのだが，このように頭を使えば使うほど，効率よく敵を倒せるのである。戦略性の存在を許すような，深みのあるシステムをもったゲームといえるのではないか。

魔王の死ぬ時

このゲームのゲームオーバーの条件は，説明するにはちょっとややこしい。スピリットが完全になくなったうえで，しかも召喚したモンスターたちも全滅するとか，敵が全員階下に逃れるとか，早くいえば手詰まりになったらおしまいなのだ。

終わってもコンティニューはできる。するとありがたいことにスピリットが回復し，終わった階からやり直しである。それを何回か続けることで，いちおうエンディングには達する。だけど，それで本当におしまいかというとそうではない気がする。やっぱりそこは，ノーコンティニュークリアを目指すべきだ。

このゲームは，モンスターが思いどおりに動いてくれなくてイラつくことはあっても，すぐ飽きてしまうということはない。なによりゲームのバランスがよーく練り込まれている。勇者たちの出現順序とか，レベルアップする経験値の設定などには職人芸的なワザさえ感じる。比較的単純なゲームシステムが，絶妙のバランスを生み出すのにひと役買っているのかもしれない。

とりあえず一度全50階を見てしまったあとは，じっくりと腰を据え，時間をかけて解くのがこのゲームの正しい遊び方だ。地味めのグラフィックと抑えた感じのBGMが，暗いダンジョンの雰囲気をよく醸し出している。固定画面の箱庭が妙なリアリズムをもってプレイヤーに語りかけてくる。

最下層に鎮座する魔王が，数々のモンスターを生み出し，攻めくる人間たちからダンジョンを守る。しかし敵を倒すには，魔王自ら骨身を削らなければならない。物量作戦は自己破滅への道でしかない。ひたすら効率よく振る舞い，経済性を追い求めるのだ。

少しでも効率を無視すると，精神力が減退していく。魔王の最期はきっとそんなふうに，持てる力の最後の一滴まで絞り出して，カスカスになって死んでいくのだ。

敵のパーティには犬もいる

オリジナルに挑戦しておくれ

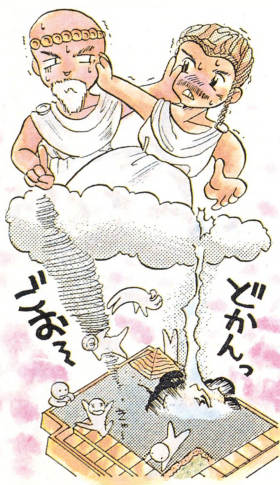
このゲームはX68000版のほかにPC-9801版も発売される。ということはこの固定画面のフィールドというのはそこらへんから来ているのだろうか。欲をいえば，このシステムでもっと広いマップを使った戦闘を楽しみたい。よく描き込まれたキャラクターどうしのパーティ戦を見てみたいのだ。ゲーム中に挿入される絵を見ていると，その方面の人材がないわけでもないらしい。もったいない話だ。

だからもし次回作があるとしたら，内容をもうグツグツに煮込んだ，X68000オリジナルの大作がいいなと思うのだ。このソフトハウスは，変に既成の作品に媚びなくて，オリジナリティがあるのがすごくいい。ソフトプランという，岡山県の会社である。

総合評価

	0　　5　　10
ゲームバランス	★★★★★★★★★
キャラクターの個性	★★★★★★
音楽	★★★★★★★
お買い得度	★★★★★★★★

AFTER REVIEW

他機種に先駆けてX68000で日本語版が発売された「ポピュラスII」。前作とはうってかわった色調の画面で，趣の異なるゲームとなりました。使える神業も増え，皆さんそれぞれの楽しみ方をしているようです。

ポピュラスII

▶止まらない。火山はムゴイ。ヘラクレスは強い。竜巻は海へ行き，渦巻は分裂する。火柱を海に沈め，沼に敵を落とす。いけーハルマゲドン。えきびょーにやられた。敵の使える技がわからない。うずうずギャー。
松本　太(21)大阪府

▶エフェクトが豪快。10MHzでも遅くない。　桝木　慎(21)愛知県

▶トロイのヘレンは強い。in 236。
紺谷　誠(21)石川県

▶よくできたパズルだと思う。
木下　孝雄(21)東京都

▶前作とはうってかわった暗い画面。
相澤　博昭(24)静岡県

▶思っていた以上に面白い。
桂川　務(18)岐阜県

▶津波や火山で，派手に敵をけちらすのが楽しい。　岡崎　恭幸(19)大阪府

▶ハードディスクに入れることはできないのでしょーか？　松岡　篤郎(21)愛知県

▶16MHzでは速すぎるというのが気に入った。どこぞの(ポリゴンとか使った)ゲームに見習わせたいものだ。
笹田　泰治(18)愛知県

▶トロイのヘレンが気に入った。
橋本　忍(21)埼玉県

▶あの「ポピュラス」をさらに面白くしたものだ。　岡部　英隆(20)奈良県

▶前作よりさらにアブナイ「ノリ」だ。
福知　健(21)京都府

▶何度やってもあきないし，奥が深い。変化があって楽しい。
町田　淳一(26)神奈川県

▶ハマっている。2台あるうちの1台のX68000はポピュラスIIのディスクをくわえ込んだまま電源も切らない。う～ん。ポピュラスはどうもオモチャっぽかったが，IIはとてもリアルで，コンピュータ相手でもたいへんマジになってやってしまっている。たまにコンピュータVSコンピュータで眺めていることもある。現在622面だが，パズル的な面などもあったり，Oh!Xで新しい技を学んで一段と楽しく勝たせてもらっている。1000面までもう少し，がんばるぞ！何度でもチャレンジできそうなゲームという気がする。対戦ポピュラスIIは，絶対ポピュラスより面白い……と私は思う。はじめからやれば経験値も関係ない。
野木　浩(32)東京都

▶ゲームは面白いが，メッセージがカタカナってのが昔の8ビットのころのゲームを思い出してしまう。青島　一高(24)静岡県

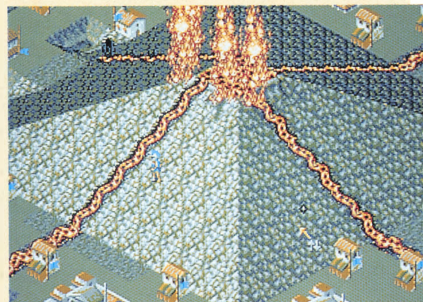
▶神業にハマる。　雨宮　光児(18)宮城県

▶やっぱり対戦が最高ですね。
山田　光一(19)新潟県

▶破壊が楽しい。　高木　誠司(17)群馬県

▶リアルタイムなところがいい。
渡部　真吾(20)愛媛県

▶コンピュータが手ごわくなっているので，対戦が使いものにならなくとも十分楽しめる。いろいろな神業が見た目もあざやかに

使えるところがよい。
後迫　浩一(31)千葉県
▶私は地獄のようなプレイヤーだ。
佐藤　真(21)愛知県
▶前作と比べてだんぜん面白い。
増﨑　達夫(28)神奈川県
▶面が多いせいか，序盤戦はサクサク勝てるので気持ちがよく中盤戦へと進めそうです。　大井　健三(40)神奈川県
▶この手のゲームはおじさんにもできますから。　大場　浩二(30)長野県
▶EXP最大の地割れはすごい。
大又　義嗣(18)大阪府
▶マウスが壊れるほど面白い。
木下　卓也(20)埼玉県
▶対戦(人対人で)すると人間関係が壊れることもある。　依田　健彦(25)東京都
▶大人のゲームになった。
石塚　潤(21)茨城県
▶あの悪夢がよみがえる(笑)。
加藤　雅浩(23)岡山県
▶あのポピュラスがこんなにパワーアップしたのか。　石田　和生(23)大阪府
▶非常に熱中でき，かつ1ゲームが短時間で終わるのでうっぷんばらしに最適！
福島　敦(20)神奈川県
▶なんだかんだいって，ハマる。
三森　浩一(23)東京都
▶マウスがつぶれそうになる？
笠野　諭(44)大阪府
▶ヒーローが面白い。
田鍋　弘司(18)広島県
▶グラフィックがすごい！
木下　義崇(18)愛知県
▶面ごとにまったく条件が違うので，あきずに遊べる。技も多彩で気分で遊べる。
植木　正幸(23)神奈川県
▶神業が，派手でGOOD。
田辺　学(21)千葉県
▶前作で，目薬をアイテムに朝日に目をしみさせていたのを君はまだ覚えているか？あの興奮をもう一度。
羽生　知浩(20)北海道
▶ストレス解消にもってこい。
原子　悟(23)北海道
▶面白さは折り紙つき。戦術・戦略を駆使して相手をねじ伏せるもよし，逃げ惑う民衆をいたぶるもよし。視覚効果は絶品。波打ち際のアニメーションが感動モノということをレビューに書き忘れたのは後悔。音関係と対戦はちょっぴり不満だけどそれ以外はほぼ完璧。移植の名手イマジニアさん，PC-9801版「F29」はアンチ98な僕をしてAMIGA版の100倍イイといわしめる出来ばえでしたが，X68000では無理だったんでしょうか？　(A.T.)

▶ポピュラスIIはなんといってもゲーム速度に満足しちゃうね。あれだけすごそうな処理をしながら10MHzのX68000でまったく支障なくプレイできるからね。さらにグラフィックがキレイ。前作の可愛らしい感じも好きだったけど，IIのリアリティあふれる感じも好きだ。魔法の種類が増えただけにちょっと操作が複雑になったけど，演出がハデだからまぁ，許す。ひとりで遊ぶゲームとしては最高峰のデキだと思うな。それだけに対戦モードがバグっているのには残念。「シムアース」のときもそうだったけど，バグ付きのソフトを見て見ぬふりして市場にほったらかしにしておくとユーザーの信用をなくすよ。あーん。対戦がしたいよ。I.I.氏をもう一度コテンパンに叩きつぶしたいよー。　(善)

発売中のソフト

★ドラゴンスレイヤー英雄伝説　SPS
X68000用　5″2HD版　9,800円(税別)
★極　ログ
X68000用　5″2HD版　12,800円(税別)
★ストライクレンジ　ブラザー工業(TAKERU)
X68000用　3.5/5″2HD版　4,800円(税込)
★Communication SX-68K　シャープ
X68000用　3.5/5″2HD版　19,800円(税別)
★KU²フロントロー　ブラザー工業(TAKERU)
X68000用　3.5/5″2HD版　2,000円(税込)
★究極タイガー　KANEKO
X68000用　5″2HD版　8,800円(税別)

新作情報

★機甲装神ヴァルカイザー　ブラザー工業(TAKERU)
X68000用　3.5/5″2HD版　4,800円(税込)
★エトワールプリンセス　エグザクト
X68000用　3.5/5″2HD版　9,800円(税別)
★沈黙の艦隊　ジー・エー・エム
X68000用　3.5/5″2HD版　12,800円(税別)
★蒼き狼と白き牝鹿・元朝秘史　光栄
X68000用　5″2HD版　9,800円(税別)
★幻影都市　ブラザー工業(TAKERU)
X68000用　3.5/5″2HD版　9,800円(税込)
★ロボスポーツ　イマジニア
X68000用　5″2HD版　価格未定
★シムアント　イマジニア
X68000用　5″2HD版　価格未定
★メガロマニア　イマジニア
X68000用　5″2HD版　価格未定
★餓狼伝説　ホームデータ
X68000用　5″2HD版　8,500円(税別)
★Traum　M.N.M Software
X68000用　5″2HD版　価格未定
★ヴェルスナーグ戦乱　ファミリーソフト
X68000用　3.5/5″2HD版　9,800円(税別)
★鮫！　鮫！　鮫！　KANEKO
X68000用　5″2HD版　価格未定
★達人　KANEKO
X68000用　5″2HD版　価格未定
★エアバスター　KANEKO
X68000用　5″2HD版　価格未定
★サバッシュII　ポプコムソフト/グローディア
X68000用　5″2HD版　価格未定
★倉庫番リベンジ/ユーザー逆襲編
シンキングラビット
X68000用　5″2HD版　6,800円(税別)
★F29 RETALIATOR　イマジニア
X68000用　5″HD版　価格未定
★マージャンクエスト(仮題)　SPS
X68000用　5″2HD版　価格未定
★麻雀悟空・天竺への道　シャノアール
X68000用　5″2HD版　9,800円(税別)

[第21回]

彼女についての記憶

寺尾　響子

響子inCGわ～るど

「さようなら……。長いあいだどうもありがとう」

そういうと彼女は扉を開けて出ていった。

たぶんこんな状況だったんだと思う。だけど，僕にはわからない。永遠に。

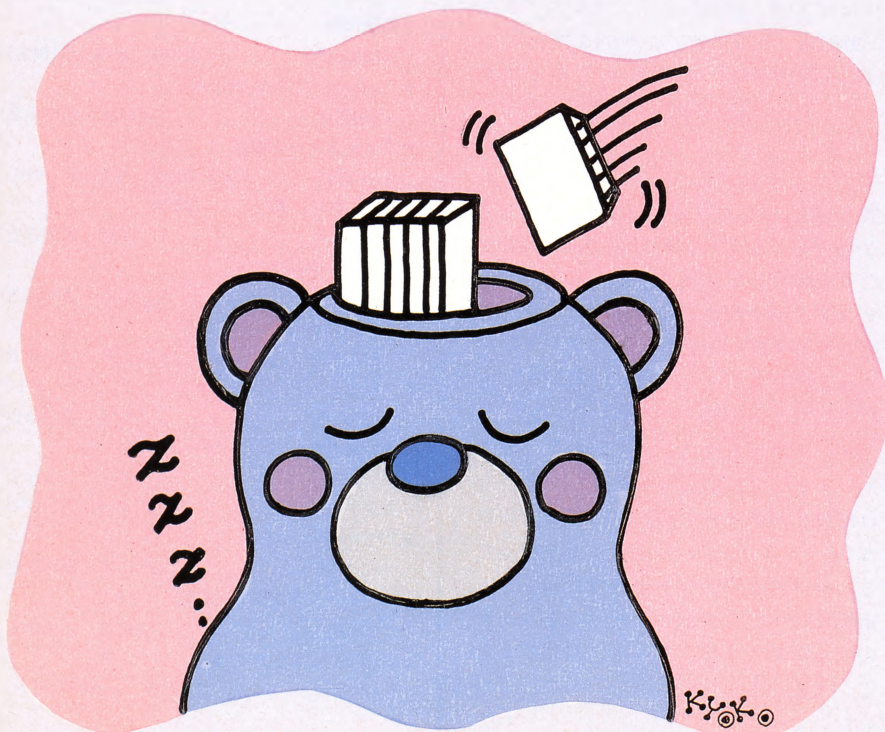
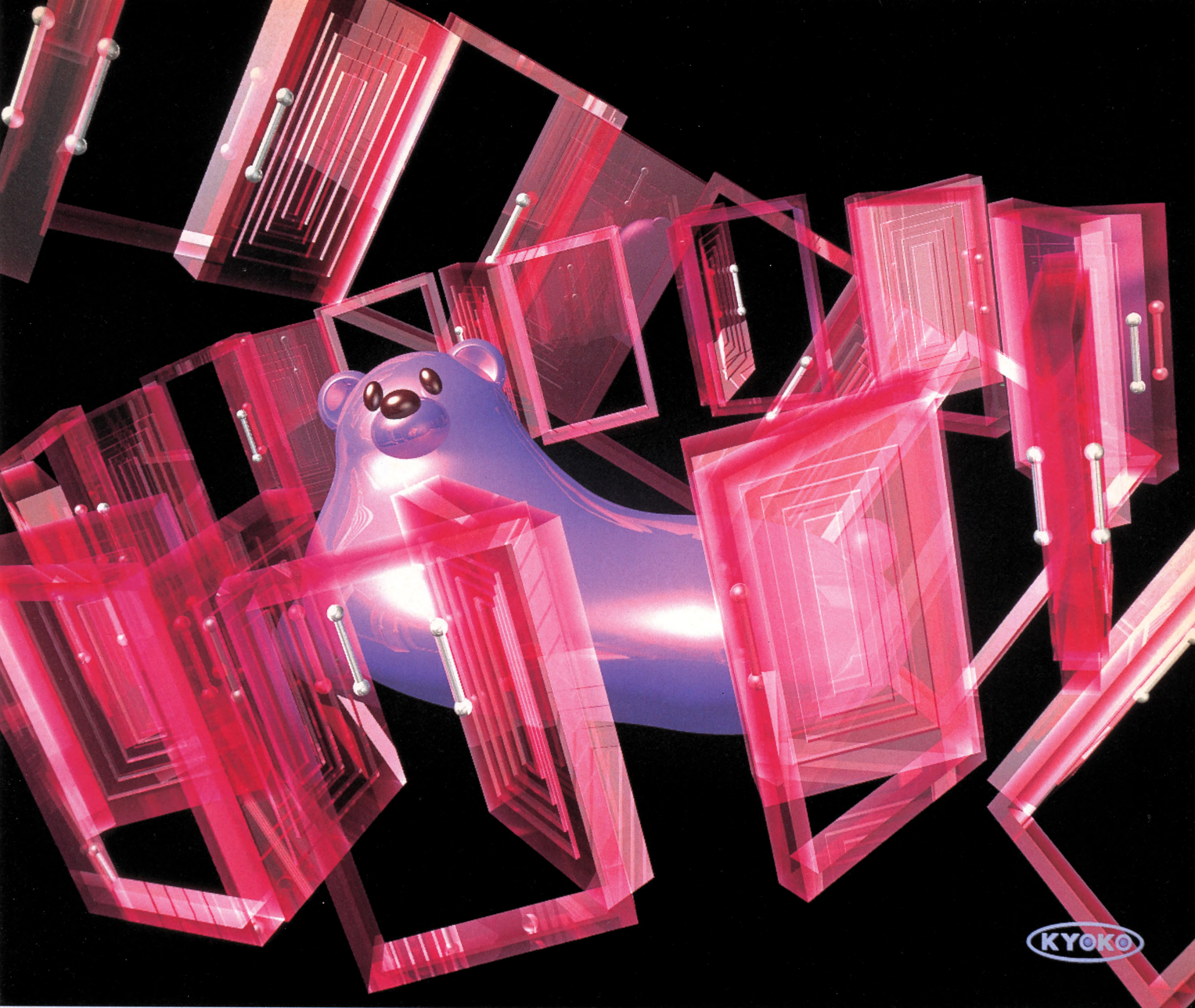
僕が彼女にしてあげられたのは，やさしく体を包んであげることだけだった。彼女が僕に求めたのもそれがすべてだった。はじめから僕と彼女はそんなふうな関係だったのだ。

女の人はだいたい気まぐれだ。それは僕もよくわかっていた。が，彼女は際立っていた。最初のうちから僕のところに来ない日が続いたのだ。僕とおなじような存在が，ほかにいくつもあるに違いなかった。彼女のことを知る機会がなかなかなくて，僕は少しあわてた。彼女が僕にどういうふうにしてほしいのかわからなかったから。でも，時がたつうちに要領がつかめてきた。

眠るときの彼女のようす。まず，背中をゆるやかな放物線のように丸める。ひざを軽く曲げて，両方のうでで抱え込む。眠りはじめはだいたい20分ごとに寝返りをうつ。深い眠りに入ると，もうほとんど動かない。動かなくてもいいように，僕が工夫したから。

ある日，いきおいよく飛び込んできて，僕に頬ずりをした。体が少し熱かった。新しいボーイフレンドでもできて，うれしかったんだろうと思う。僕は彼女がゆっくりと楽しい夢を見られるようにしてあげた。

また，こういうときもあった。静かに横になって，しばらく動かなかった。それから，彼女の肩が震えた。よくわからなかったが，たぶん泣いていたんだろう。僕は彼女の体をそっと包んだ。疲れが癒えるようにと。

こんなふうにして，僕はひととおり彼女のことを思い出した。細かな感触も含めて，ぜんぶ。

それから，ていねいに彼女についての記憶を圧縮すると，まだ空いているメモリの片隅にそっと置いた。暗いメモリ空間には，いくつもの記憶が並んでいた。彼女のまえの彼女，その彼女のまえの彼女，そのまえの彼女のまえの彼女のまえの彼女。ずっと続く彼女たちに関する記憶。

だけど，僕は，僕は，僕は，僕は，僕は……。

＊　＊　＊

人間に，彼の思いは伝わりませんでした。なぜなら，彼にはことばを発する器官がなかったからです。彼はただのウォーターベッドでした。

彼の表面はさざなみのように細かく変形し，そして大きくうねりました。内部の温度が人肌よりも高く上昇しました。それも一瞬のあいだで，半秒ののちに温度はぐんぐんと下がり，金属のように冷たくなっていきました。彼自身が眠りに入ったのです。

数日して，首都圏住宅情報ネットワークに新しいデータが加わりました。

貸室あり
千葉県浦安市　8帖　ワンルーム　314階
賃貸期間2年　女子大生にかぎる
クマのぬいぐるみ型
AIコントロールウォーターベッドつき

Communication SX-68K

Taki Yasushi 瀧　康史

多くのネットワーカー&SX-WINDOWファンの期待に添うべく，「Communication SX-68K」が発売されました。SX-WINDOWの機能を生かし，使い勝手のいいアプリケーションに仕上がっているのでしょうか？

現在はフリーウェアの通信ソフトが多く出回っています。普通に考えるのなら，質のいい通信ソフトがアクセス料と電話代で手に入る環境でソフトを発売するには，それ以上の性能を背負わなくてはならないのは当たり前のことです。最初にフリーウェアの通信ソフトをダウンするために使ってもらおう，なんて甘いことを考えているわけではないでしょうからね。

まずは操作性

イージーであるか，マニアックであるかは別としても，利用していて苦痛を感じさせないかどうかは，優良ソフトへの道の最初の関門といえるでしょう。

先月のSOUND SX-68Kと同じく，アイコンの登録などはすべてオートです。システムも当然入っていないようです。新しく作られたRSDRV.SYSを，デバイスドライバに追加することからインストールは始まります。ほかのRS-232Cドライバは削除しておきましょう。

起動してみると，ほんのわずかの間だけ，ログウィンドウにタイトルが表示されます。上のほうに電話アイコンとか，ディスクアイコンとか，環境アイコンがあって，見た目はエディタ.Xに似ています。

やることはまず環境の設定で，通信ソフトらしいありきたりの設定をします。新JISなどの設定やNECJISとかもあり，通常の使用では困らないでしょう。最大レートは19200bps。通信ソフトによっては38400bpsがあるものもありますが，まあ，マルチタスクで動くんですから，これはいいことにしましょうか。

通信条件などの設定はダイアログで

X68000用　3.5/5"2HD版　19,800円(税別)
シャープ　☎03(3260)1161

SX-WINDOWの上で動くのですから，当然ウィンドウのリサイズやフォントの選択もできます。書体倶楽部の新明朝体を愛用している私としては悲しいのですが，ベクトルフォントは使えません。まあ，速度の問題があるからでしょう。

ダウンロード先やアップロード先も当然のように指定できます。流行の1行入力にもちゃんとヒストリがついていますし，“opt.1”キーでアクティベートせずに，メニューを引き出したり（ゆえにXMODEMをダウンしたり）できます。速度面でもソコソコのスピードは出ていますし，バックログを見るのも簡単。SX-WINDOWアプリとしても，通信ソフトとしても，標準的な機能は常備しています。と，ひと言でいってしまうのは簡単ですが，当たり前の機能を持っていないアプリケーションをX68000ユーザーなら何度も見てきたでしょう。

そしてログイン

あらかじめ有名なBBSやネットは，登録されているので，とりあえず自動ログインを実行してみます。スクロールバーで選択できるなどと，SX-WINDOWらしい機能がなかなかオシャレかな。

そのまま実行すると，ID&パスワードが全部XXXXXXになってしまうので，自分の通っているネットだけ修正します。

オートログインの設定はほとんどすべてメニュー選択なので，たいして時間もかからずに，ぽいぽいと作成できます。エディタ.Xのお世話にはあまりならないでしょう。初期設定もほとんどモデムのメーカーを選ぶだけですから，たいして手間はかかりません。初心者でもたぶんOK。ただし，この作業をすべてダイアログでやらねばならないというのはちょっと面倒です。

アイコンも自動的に登録される

当然SX-WINDOWですから，オートログインやダウンロード中はマルチタスクします。しかし，速度の問題かどうかは知りませんが，XMODEMなどのバイナリ通信中はダイアログが表示されてしまいます。このダイアログにはアップロードまでにかかる予想時間や，全体のバイト数，いままで転送が終わっているバイト数，ブロック数など必要な情報がすべて明記されていて便利なことは便利ですが。

エディタなどで作成しているデータをそのままクリップして，転送することもできますし，バックログから転送することもできます。

総合するとかなり，ポイントは高いでしょう。妙な不都合もないようですし。

贅沢をいえば，オートパイロットなどをより充実させてほしいところですが，そういったことよりもウィンドウ環境での通信ソフトの便利さのほうが，ポイントが高いんじゃないでしょうか？

確かに，クイックVANやBPLUSのような，大手のバイナリ転送もサポートしてほしかったところですが，ファイル形式については外部ファイル（当然SX-WINDOW用）を参照できるようになっていて，ドキュメントもついているので，誰か自作してくれる人がいるでしょう。

まだまだいいたいことはあるのですが，とりあえず私は，このソフトに乗り換えてみようかと思います。

リソースを使ってみる

久しぶりに復帰したSX-WINDOW講座です。いよいよ大詰め，リソースマン関係の関数を扱います。リソースという概念をうまく把握できるかがポイントです。サンプルプログラムは神経衰弱ゲームです。

Nakamori Akira 中森 章

先日，自動車免許を取ったばかりの友人に川崎から熱川のバナナワニ園までドライブにつきあわされました。そこで見た白いワニのせい（私は江口寿史か？）ではありませんが，この連載をたて続けに休んでしまいました。本当にごめんなさい。

さて，今回はリソースを扱ってみようと思います。リソースはまじめにやると非常に奥の深い概念ですが，説明するのが私ですからそんなに厳密にはやりません（やれません）。リソースファイルをオープンしてその中のリソースを参照するやり方を覚えたら，さっさとサンプルプログラムに移るつもりです。その程度の知識があれば実用上は十分だと思います。それでは始めましょう。

リソースとは

リソースとはそのまま日本語に訳せば「資源」という意味です。謹賀新年PRO-68Kの付録ディスクに収録されていたSX-WINDOWのドキュメントのリソースの説明は，「Macintoshのリソースとはまったく別物」という文で始まっています。この1文でSX-WINDOWが（リソース以外は）Macintoshの環境を真似て作られていることをうかがい知ることができます。しかし，この説明でリソースがなんなのかわかる人はいないと思います。「Macintoshと同じ」ならともかく，「Macintoshとは別物」といわれて内容を想像できるわけがありません。

SX本（参考文献1））によると，Macintoshではリソースがすべての基本で，それを理解しないと話にならないほどの重要な概念ですが，SX-WINDOWではそれほどの重みを持ってないということです。ああよかった。

謹賀新年PRO-68Kのドキュメントから SX-WINDOWでのリソースの説明を抜粋すると次のようなものになります。

●リソースはファイルの特殊な形式

●リソースはタイプとIDを持ち，それで個々を識別する

●タイプは32ビット整数で，通常ASCIIコードの4文字で表す

●IDは16ビットの符号付き整数で表す

●メッセージやグラフィックデータはプログラムのコード中に埋め込まず，リソースとしてコードから分離するのが理想

●リソースをいくつか集めたものがリソースファイルで，それはHuman68k上のファイルになる

でも，これはリソースというものをわざわざ難しく説明しているような気がします。本当はもっと簡単なのではないでしょうか。

いろいろなプログラム例を見ると，SX-WINDOWにおけるリソースとは，プログラムコードから分離されたサブルーチンやデータのこととわかります。そのサブルーチンやデータはハンドル（例のポインタへのポインタというやつ）を通じてプログラムから参照されます。ハンドルで参照しますから，リソースとして定義されるサブルーチンは当然リロケータブルな構造になっていなければなりません。また，これらのサブルーチンやデータを寄せ集めてファイルにしたものがリソースファイルです。リソースファイルのファイル名は（Human68kが認識できるものなら）なんでもいいのですが，慣例的に拡張子は.LBにすることが多いようです。

ところで，リソースの存在意義はなんでしょう。一般には次のようなことがよくいわれます。

「プログラムで表示するメッセージなどを変更したいときに，メッセージ部分がリソースとしてプログラムから分離されていれば，プログラムを変更しなくてもリソースを変更するだけでよい。たとえば，異なる言語にも容易に対応できる」

誰がいい始めたのか知りませんが，こんな些細な理由でプログラムとリソースを分けなければならないとしたら非常に馬鹿げていると思います。この説明はプログラムを作る側の論理であって，プログラムを使う側には無意味です。プログラムを使う側にとっては，プログラムとリソースが分かれることでファイル数が増加するのが，かえってうっとうしく思えるでしょう。

それに，プログラムを作る側にとっても，リソースを作り直す暇があるなら，プログラム自体を作り直しても手間は変わらないと思います。ですから，私は「ユーザーの目に触れるものをすべてリソースにするのが理想」という考えには反対です。ユーザーがカスタマイズする余地のある部分だけリソースにしておけばよいのではないでしょうか（ユーザー側がプログラムを作り直すのは困難ですから）。

私はリソースにはもっと別の意義があると思っています。それは資源の共通化です。たとえば，トランプや花札のカードのグラフィックデータなど，どのアプリケーションでも変わりようのない（膨大な量の）データをアプリケーションごとに持っているのは不合理です。そのようなデータこそリソースにして，いろいろなアプリケーションで共通のリソースファイルを参照するようにすればよいでしょう。そうすることによって，アプリケーションごとに同じようなデータを持つ必要がなくなるので，ディスク容量の節約になります。

もっとも，あとで説明するように，リソースファイルはアプリケーションごとにメモリに読み込まれますから，リソースファイルを共通化したからといってメモリ容量の節約にはなりませんが（うまくやればメモリも節約する方法があるかもしれない）。

リソースファイルの構造

リソースを集めたものがリソースファイルです。リソースファイルは大きく分けて，リソースマップとリソースそのもので構成されています。リソースマップは，どのタ

イプでどのIDを持ったリソースが，リソースファイルのどのあたりに格納されているかを示すマップ（地図）です。

アプリケーションがリソースファイルをメモリに読み込む場合，とりあえずこのリソースマップのみが読み込まれます。そして，リソースの実体は，アプリケーションが要求したときに初めて，リソースマップを頼りに，メモリに読み込まれるような仕組みになっています。

リソースファイルに含まれるリソースをいつもすべて利用する場合ばかりとも限らないので，メモリ効率を考えてそういう仕組みになっているのでしょう。しかし，多くの場合，特に独自に作ったリソースファイルでは，リソースファイル内のリソースをすべてメモリに読み込みます。そういった場合は，アプリケーションの要求ごとにリソースをひとつずつメモリに読み込むほうが効率が悪いので，すべてのリソースをあらかじめメモリに読み込むという処理を行うこともできます。

リソースマップのイメージを具体的なものにするために，リソースファイルの構造を図1に示します。リソースマップとは図のリソースファイルの「リソースの実体」以外の部分です。この部分がメモリに読み込まれて展開されます。

そして，個々のID情報の「リソース実体へのハンドル」の部分が0に初期化されます。この部分が0ならば，SX-WINDOWのシステムは，リソースの実体がまだメモリに読み込まれていないと判断するわけです。

リソースファイルの構造がわかっていればリソースファイルを直接作り出すのは簡単です。ただし，リソースやリソースファイルの構造は将来変更になる可能性があるので，リソースファイルの作成はSX-WINDOWのシステムコールを利用して行うのがよいでしょう。しかし，そういうプログラムをわざわざ作成しなくても，Human68kのコマンドラインからリソースファイルを作るツールがすでにあります。謹賀新年PRO-68Kの付録ディスクに収録されたRLK.Xや，追補版SX本（参考文献2））の付録ディスクに収録されたフリーウェアのRSC.Xがそれです。これらのツールを使用すれば，

RLK -T<タイプ> -I<ID> <リソースファイル> <リソース>

または，

RSC -a -T<タイプ> -I<ID> <リソースファイル> <リソース>

によってリソースファイルを作成したり，新しいリソースを追加することができます。

図1　リソースファイルの構造

リソースを扱う関数

それでは，リソースファイルがすでに存在するものとして，SX-WINDOWでリソースを扱う方法について説明しましょう。表1にSX-WINDOWでリソースを扱うための主な関数を示します。リソースを扱うマネージャはリソースマンと呼ばれ，表1の関数はすべてリソースマンの関数です。

表1にはリソースマップに対してリソースを追加/削除する関数も載せてありますが，実際にはリソースマップにあるリソースをただ参照する以外の使用法は少ないと思われます。一般的には，だいたい次のような手順でリソースを参照します。

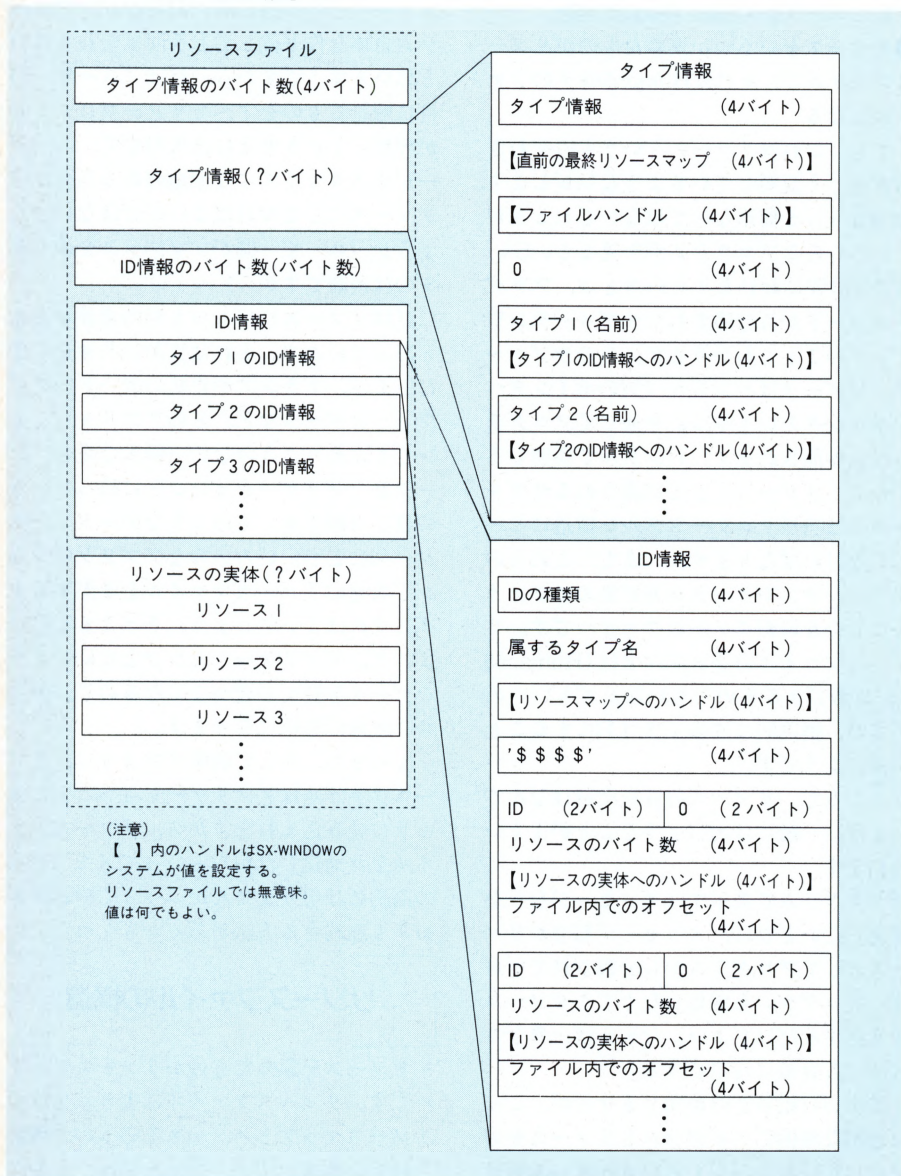
1)　リソースファイルのオープン

最初はアプリケーションプログラムが必要とするリソースが格納されているリソースファイルをオープンします。このための関数が，

RMResOpen

です。この関数はリソースファイルの中からリソースマップの部分をメモリに読み込み，そのリソースマップへのハンドルを値として返します。リソースファイルは読み込み専用モードでオープンされ，ファイル自体はオープンされたままになっています(あとからリソースの実体を読むため)。

また，メモリに読み込まれたリソースマップは，同時に，最終リソースマップ，およびカレントリソースマップ（へのハンドル）としてシステムに登録されます(図2)。

最終リソースマップとは最後にオープンされたリソースファイルのリソースマップ

です。リソースマップのデータ構造にはひとつ前の最終リソースマップへのハンドルを格納する領域があります(図1参照)。現在の最終リソースマップからひとつ前の最終リソースマップの情報を順番にたどっていくことで，いままでにオープンされたすべてのリソースマップを参照することができます。

カレントリソースマップはリソースに対する処理を行う場合に，基準となるリソースマップです。リソースマンのほとんどすべての関数はカレントリソースに対する操作を行います。また，リソースを検索する場合もカレントリソースを始点として，それよりも古いリソースマップの中を探しにいきます。

2) リソースのメモリへの読み込み

通常は，アプリケーションプログラムがリソースを要求して初めて，つまりそれぞれの要求ごとに，リソースの実体がメモリに読み込まれます。しかし，リソースファイルの内容をすべて読み込む必要がある場合には，最初に一括して読み込んでおいたほうが効率的です。このための関数が，

RMResLoad

です。この関数はカレントリソースマップのリソースをすべてメモリに読み込み，リソースファイルをクローズします。また，リソースマップの中のいくつかのリソースを要求したあとで，この関数を呼び出すと，まだメモリに読み込まれていないリソースのみが読み込まれることになります。

3) リソースの要求

アプリケーションプログラムが参照するリソースを要求する場合は，タイプとIDで指定します。タイプは通常4バイトの整数で，上位から8ビットずつを文字コードとみなして4文字の文字データとして扱います。C言語で記述するときは4つの文字を，

'CODE' ⟷ 0x434F4445
'PAT4' ⟷ 0x50415434
'DLOG' ⟷ 0x444C4F47

のようにシングルクオートで囲みます。なお，タイプのうち'　　'(0x20202020)以下のものはシステム予約になっています。

また理論的には3文字以下の名前のタイプも可能ですが，この場合は先のRLK.XとRSC.Xでリソースファイルへの名前の登録方法が異なっており(というよりもRSC.Xでは3文字以下のタイプを考慮してない)，互換性がないので使用しないほうがよいでしょう。IDは16ビットの整数で，

－32768～32767

の範囲を取ります。一応，127以下(負を含む)のIDはシステム予約になっていますが，タイプの指定さえ間違えなければそれほど神経質になる必要はないと思います。

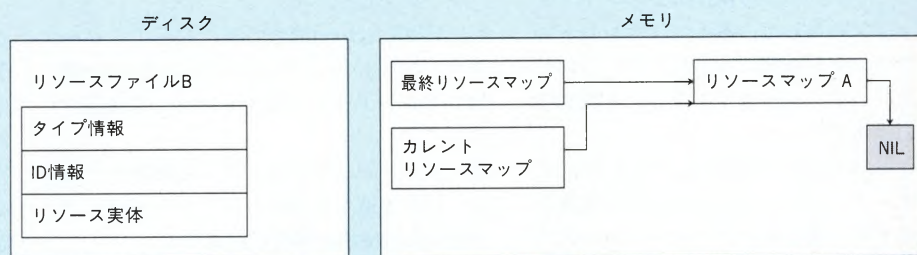
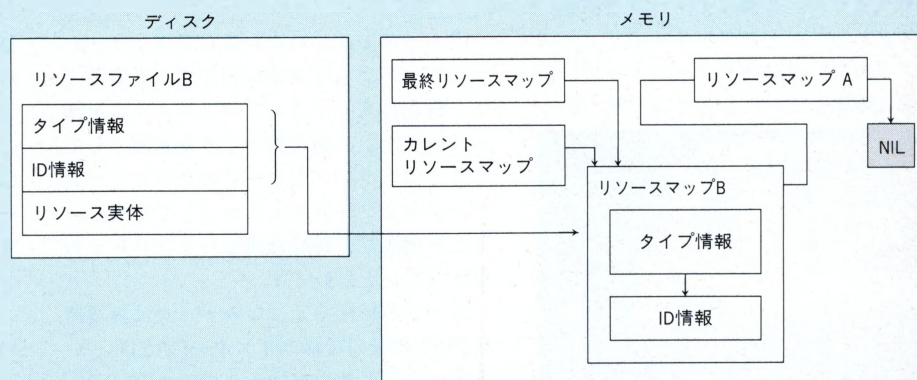
リソースの検索を行う関数は，

RMRscGet

です。この関数はカレントリソースマップよりも過去にオープンされたリソースファイルのリソースマップの中に，指定されたタイプとIDに一致するリソースを探しに行き，最初に見つかったリソース(の実体)

表1　リソースを扱う主な関数

関数	説明
ファイルの操作	
handle RMResOpen(char ＊name)	nameで指定するリソースファイルをオープンし，カレントリソースマップにする
int RMResLoad(void)	カレントリソースマップに属するリソースをすべてメモリ上に置く
int RMResClose(char ＊name)	カレントリソースマップをnameで指定するリソースファイルにセーブして削除する
int RMResRemove(void)	カレントリソースマップをファイルにセーブしないで削除する
カレントリソースマップへの操作	
handle RMCurResGet(void)	カレントリソースマップへのハンドルを返す
handle RMCurResSet(handle newRes)	newResをカレントリソースマップに設定する
int RMTypeRemove(long type)	カレントリソースマップのtypeで指定されるすべてのリソースを削除する
handle RMRscAdd(long type, int id, handle hdl, int size)	カレントリソースマップに新しいリソースを加える
リソースマップ全体への操作	
handle RMRscGet(long type, int id)	カレントリソースマップから探して，typeとidで示されるリソースを返す
int RMRscRemove(long type, int id)	typeとidを指定し，リソースをカレントリソースマップから削除する

図2　リソースファイルのオープン

へのハンドルを返します。

4)　リソースファイルのクローズ

アプリケーションプログラムの終了時など，メモリに読み込んだリソースマップが不要になったら，リソースマップを廃棄しリソースファイルをクローズしなければなりません。このための関数が，

　RMResRemove

および，

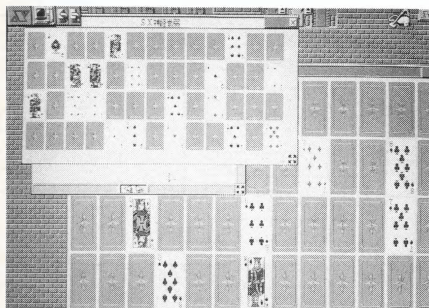
　RMResClose

です。

RMResRemove関数はカレントリソースマップをメモリから削除し，カレントリソースマップを，その次にオープンしたリソースマップに変更します。またオープンしていたリソースファイルをクローズします。一方，RMResClose関数はカレントリソースマップの内容をRMResLoad関数でいったんメモリに読み込んだあと，引数で与えられたファイル名でディスクにセーブし，RMResRemove関数を呼び出します。カレントリソースマップの内容をそのままセーブしますから，カレントリソースマップに追加や変更があった場合はそれらも同時にセーブされます。通常のアプリケーションプログラムではリソースマップを変更するということはないので，RMResClose関数を使用することはあまりないでしょう。

＊　＊　＊

以上の説明を見てわかるように，リソースマップやリソースの操作はカレントリソースマップに対して，あるいはカレントリソースマップを基準に行われます。このため，カレントリソースマップという概念は非常に大切です。

リソースのタイプやIDというものはアプリケーションプログラムが勝手に定義するものです。異なるアプリケーションプログラム間で，内容が違うのに同じタイプで同じIDのリソースが存在しないという保証はありません。必要なリソースを最初にオープンしたリソースファイルのリソースマップ（あるいは最終リソースマップ）から順番に検索していったのでは，見つかったリソースが本当に必要なリソースか否かが判別できません。

大きさ自由自在

アプリケーションプログラムが必要なリソースは，通常アプリケーションプログラム自身がオープンしたリソースファイルの中にあります。したがって，リソースは自分自身のリソースマップから探し始めなければなりません。こうすれば最初に見つかるリソースが本当に必要なリソースであることが保証されます。

このように，個々のアプリケーションプログラムがリソースを検索する始点や操作対象となるプログラムを決定するためにカレントリソースマップというものが存在します。このため，アプリケーションプログラムはリソースを操作や検索するたびにカレントリソースマップをこまめに切り替えてやる必要があります。そのための関数が，

　RMCurResGet

および，

　RMCurResSet

です。

RMCurResGet関数はカレントリソースマップへのハンドルを得る関数，RMCurResSet関数はリソースマップへのハンドルをカレントリソースマップとして設定する関数です。つまり，リソースマンの関数を呼び出すための一般的な操作は次のようになります。

1)　RMCurResGet関数でカレントリソースマップを退避する。

2)　RMResOpen関数で得た自分のリソースマップへのハンドルをRMCurResSet関数でカレントリソースマップとして設定する。

3)　カレントリソースに対する操作をする。

4)　1)で退避したカレントリソースマップをRMCurResSet関数で回復する。

サンプルプログラム

リソースとは，いろいろなアプリケーションプログラムで共通に利用できる資源という観点に立ってサンプルプログラムを作ってみます。共通に利用できる資源として，ここではトランプのカードのグラフィックデータをリソースファイルとして作成し，それを使用して神経衰弱を行うプログラムを作りたいと思います。

カードの絵柄の元になるデータは謹賀新年PRO-68Kの付録ディスクのCARD2というディレクトリに格納されているCARDDRV.X用のグラフィックデータ（ファイル名はTR.DAT）を使用します。それを先に図1に示したリソースファイルの形式に直接変換することにしましょう。そのためのプログラムがリスト1です。

カレントディレクトリにTR.DATを置いて，リスト1のプログラムを実行すると，カレントディレクトリにCARDS.LBというリソースファイルができあがります。このリソースファイルにはトランプの絵柄が，タイプが'CARD'，IDが0から53で格納されています（システム予約のIDを使ってしまった）。意味をはっきりさせるために'CARD'という名前になっていますが，データ構造自体は，4画面を使用するテキストタイプのビットイメージである'PAT4'と同じものです。なお，IDとカードの絵柄との関係は次のようになっています。

0	トランプの裏側
1～13	スペードの1～13
14～26	ハートの1～13
27～39	ダイヤの1～13
40～52	クラブの1～13
53	ジョーカー

ところで，もしリソースファイルではなく，それぞれのリソースのソースファイルがほしいときはリスト1の先頭にある，

　#define BINARY

の1行を削除してコンパイルして実行してください。カレントディレクトリに，

　CARD0.S　～　CARD53.S

という54個のファイルが作られます。

そして，神経衰弱のプログラムはリスト2です。ウィンドウに表示されたカードのうち，裏返しになっているものをダブルクリックすると表になります。2枚のカードを開いてカードの数字が一致したら，カードは開いたままになります。もし，カードの数字が不一致なら，次のダブルクリック時に新しいカードを開くと同時に，以前の2枚のカードが裏返しになるようになっています（写真参照）。この処理（270～320行目のあたり）が少し複雑ですが，SX-WINDOWのマネージャに関する処理ではないので説明は省きます。

それを除けば，プログラム自体は大したことありません。すでに説明したようにCARDS.LBというリソースファイルをオープンして，タイプが'CARD'でIDが0および1から52のリソースを取り出してウィンドウ上に表示しているだけです。

ただ，ウィンドウの内部いっぱいに52枚のカードが並べられるようにリソースをウィンドウのサイズに応じて拡大/縮小しながら表示する部分がSX-WINDOWのマネージャに関する処理といえるかもしれませ

ん。しかし，それはこの連載の第9回で説明したGMTransImg関数の応用ですからわかりますよね。まあ，全体にリスト内のコメントを読んでもらえば処理の流れは理解できるでしょう。

なお，リソースファイルがカレントディレクトリにない場合を考慮して，TSSearchFile関数でリソースファイルの位置を検索するようにしてあります。これは，ファイルをオープンする前の約束ごととして覚えておきましょう。

＊　　＊　　＊

今回で説明をしたかったSX-WINDOWのマネージャの大部分が終了しました。あと，テキストマン，サブウィンドウマン，プリントマン，メモリマンなどが残っていますが，私はあまり面白みを感じません。それらについては機会があれば（その気になったら）説明したいと思います。来月からは，少し基本に立ち返ってウィンドウマンあたりから復習をしてみましょう。

《参考文献》

1）　吉沢正敏，SX-WINDOWプログラミング，ソフトバンク，1991年.

2）　吉沢正敏，追補版SX-WINDOWプログラミング，ソフトバンク，1991年.

3）　謹賀新年PRO-68K，Oh!X1992年1月号付録.

リスト1

```
===============  SX1  ==================
  1: /*
  2:      CARDFUNC のデータを リソースファイルに変換する
  3:      あるいは PAT4 形式のソースに変換する
  4:
  5:      謹賀新年PRO-68K(1991年1月号付録)の
  6:      CARD2ディレクトリにあるTR.DATを変換
  7:
  8:                     中森 章 Dec. 5, 1992
  9: */
 10: #include <stdio.h>
 11: #define BINARY
 12:
 13: char    card[96][48];
 14: char    p[4][96][48];
 15: int col[]={0,3,-1,7,-1,5,-1,-1,-1,-1,-1,-1,6,4,-1,8};
 16: #ifdef BINARY
 17: int hd[]={
 18:   0x18, 1, 0, 0, 0, 'CARD', 0, 0x370, 0x36, 'CARD', 0, '$$$$'
 19: };
 20: #endif
 21: main()
 22: {
 23:     FILE    *fpr,*fpw;
 24:     int     x,y,z;
 25:     int     ch;
 26:     int     i,j,k;
 27:     short   s;
 28:     char    fn[90];
 29:     fpr=fopen("tr.dat","r");
 30:     if(fpr==NULL){
 31:         fprintf(stderr,"TR.DATがオープンできません。¥n");
 32:         exit(1);
 33:     }
 34:     fseek(fpr,0x210,0);
 35: #ifdef BINARY
 36:         fpw=fopen("CARDS.LB","wb");
 37:         if(fpw==NULL){
 38:             fprintf(stderr,"ライト用ファイルがオープンできません。¥n");
 39:             exit(1);
 40:         }
 41:         fwrite((char*)hd,4,12,fpw); /* ヘッダを書く */
 42:         for(i=0,j=0x390;i<54;i++,j+=0x908){
 43:             k=i<<16; fwrite((char*)&k,4,1,fpw);
 44:             k=0x908; fwrite((char*)&k,4,1,fpw);
 45:             k=0;     fwrite((char*)&k,4,1,fpw);
 46:                      fwrite((char*)&j,4,1,fpw);
 47:         }
 48: #endif
 49:    for(z=0;z<54;z++){
 50:         for(y=0;y<96;y++){
 51:             for(x=0;x<48;x+=2){
 52:                 ch=fgetc(fpr);
 53:                 card[y][x+0]=(ch>>4)&0xf;
 54:                 card[y][x+1]=(ch)&0xf;
 55:             }
 56:         }
 57: #ifndef BINARY
 58:         sprintf(fn,"CARD%02d.S¥0",z);
 59:         fpw=fopen(fn,"w");
 60:         if(fpw==NULL){
 61:             fprintf(stderr,"ライト用ファイルがオープンできません。¥n");
 62:             exit(1);
 63:         }
 64:         fprintf(fpw,"CARD_%02d:¥n",z);
 65:         fprintf(fpw,"¥t.dc.w¥t0,0,48,96¥n");
 66: #else
 67:         k=0;          fwrite((char*)&k,4,1,fpw);
 68:         k=0x00300060; fwrite((char*)&k,4,1,fpw);
 69: #endif
 70:         for(y=0;y<96;y++){
 71:             for(x=0;x<48;x++){
 72:                 ch=col[card[y][x]];
 73:                 p[0][y][x]=(ch&0x1)? 1:0;
 74:                 p[1][y][x]=(ch&0x2)? 1:0;
 75:                 p[2][y][x]=(ch&0x4)? 1:0;
 76:                 p[3][y][x]=1;
 77:             }
 78:         }
 79:         for(i=0;i<4;i++){
 80:             for(y=0;y<96;y++){
 81: #ifndef BINARY
 82:                 fprintf(fpw,"¥t.dc.w¥t%%");
 83: #else
 84:                 s=0;
 85: #endif
 86:                 for(x=0;x<16;x++){
 87: #ifndef BINARY
 88:                     fprintf(fpw,"%x",p[i][y][x]);
 89: #else
 90:                     s=s+s; if(p[i][y][x]) s++;
 91: #endif
 92:                 }
 93: #ifndef BINARY
 94:                 fprintf(fpw,",%%");
 95: #else
 96:                 fwrite((char*)&s,2,1,fpw);
 97:                 s=0;
 98: #endif
 99:                 for(x=16;x<32;x++){
100: #ifndef BINARY
101:                     fprintf(fpw,"%x",p[i][y][x]);
102: #else
103:                     s=s+s; if(p[i][y][x]) s++;
104: #endif
105:                 }
106: #ifndef BINARY
107:                 fprintf(fpw,",%%");
108: #else
109:                 fwrite((char*)&s,2,1,fpw);
110:                 s=0;
111: #endif
112:                 for(x=32;x<48;x++){
113: #ifndef BINARY
114:                     fprintf(fpw,"%x",p[i][y][x]);
115: #else
116:                     s=s+s; if(p[i][y][x]) s++;
117: #endif
118:                 }
119: #ifndef BINARY
120:                 fprintf(fpw,"¥n");
121: #else
122:                 fwrite((char*)&s,2,1,fpw);
123:                 s=0;
124: #endif
125:             }
126: #ifndef BINARY
127:             fprintf(fpw,"*¥n");
128: #endif
129:         }
130: #ifndef BINARY
131:         fclose(fpw);
132: #endif
133:     }
134:     fclose(fpr);
135: #ifdef BINARY
136:         fclose(fpw);
137: #endif
138: }
```

リスト2

```
===============  SX2  ==================
  1: /*
  2:
  3:     SX-WINDOW  リソースファイルのサンプル
  4:
  5:                         (C) 中森 章, Dec. 6, 1992
  6: */
  7: #include <stdio.h>
  8: #define __POINT_T    /* point_t 型を使う */
  9: #include <sxlib.h>
 10: #define FALSE   0
 11: #define TRUE    ~FALSE
 12: /*
 13:     ここでウィンドウに関する定数を設定
 14: */
 15: #define WDEFID      WI_STD
 16: #define WINOPT      ( WC_GBOX | WC_GBOXON )
 17: #define WINWIDTH    700
 18: #define WINHIGHT    400
 19: #define WINTITLE    "¥014SX神経衰弱"
 20: #define EVENTMASK   EM_EVERY
 21: /*
 22:     ここは定数から計算される定数
 23: */
 24: #define WINOPTL     ( WINOPT & 0xf )
 25: #define WINDEFID    ( WDEFID << 4 | WINOPTL )
 26:
 27: window  *winPtr;
 28: rect    winSize;
 29: event   eventRec;
```

```
 30: int      activeFlag;
 31: int      ctrlFlag;  /* コントロールがあるかないか      */
 32: int      menuFlag;  /* メニューがあるかないか            */
 33: menu     **menuHdl; /* ポップアップメニューのハンドル */
 34: int      lastWhen;  /* ダブルクリック識別用               */
 35:
 36: #ifdef __GNUC__
 37: asm( ".xdef _STACK_SIZE"  );
 38: asm( "_STACK_SIZE equu   8192" );
 39: asm( ".xdef _HEAP_SIZE"   );
 40: asm( "_HEAP_SIZE equ    16384" );
 41: #endif
 42:
 43: handle  myResFileHdl;     /* リソースファイルのハンドル */
 44: char    myResFileNam[90];/* リソースファイルの名前       */
 45: int     resFlag;          /* リソースファイルがオープンできたか? */
 46: point_t cardxy0,cardxy1; /* カード座標計算用の原点        */
 47: int     cardx,cardy;      /* カードの縦横の大きさ          */
 48: char    CARD[4][13];      /* カード位置の記憶用            */
 49: int     cardopen;         /* -1: カードをまだめくってない    */
 50:                           /*      あるいは番号が一致している */
 51:                           /*  0: 一枚目のカードをめくった    */
 52:                           /*  1: 二枚目のカードをめくった    */
 53: int     cardEven_x;       /* 一枚目のカードのX座標          */
 54: int     cardEven_y;       /* 一枚目のカードのY座標          */
 55: int     cardEven_id;      /* 一枚目のカードの番号            */
 56: int     cardOdd_x;        /* 二枚目のカードのX座標          */
 57: int     cardOdd_y;        /* 二枚目のカードのY座標          */
 58: int     cardOdd_id;       /* 二枚目のカードの番号            */
 59: /*
 60:      (0) リソースファイル名をつくる
 61: */
 62: ResFileName(char fname[])
 63: {
 64:     task     taskBuf;
 65:     int      len;
 66:
 67: #if 0    /* 普通はこんな感じでファイル名を決めるのかな */
 68:
 69:     TSGetTdb(&taskBuf, -1);
 70:     strcpy(fname, taskBuf.name); /* 自分のファイル名 */
 71:     len=strlen(fname);
 72:     while((len>0) && (fname[len]!='.')) len--;
 73:          /* 後ろから探して '.' の位置を調べる */
 74:     fname[len+1]='L';
 75:     fname[len+2]='B';
 76:     fname[len+3]=0; /* 拡張子を '.LB' にする */
 77: #else
 78:     strcpy(fname,"CARDS.LB");    /* 一意な名前 */
 79: #endif
 80:
 81:     return 0;
 82: }
 83:
 84: /*
 85:      (1) リソースファイルのオープン
 86: */
 87: OpenMyRes(handle *myRes, char *fname, int load)
 88: {
 89:     handle  curRes;
 90:     char    dname[90];
 91:
 92:     curRes = RMCurResGet(); /* カレントリソースを退避 */
 93:     if(TSSearchFile(fname, dname, ".")<0) return -1;
 94:                             /* 新たなリソースファイルを検索 */
 95:     if( (*myRes = RMResOpen(dname)) <(handle)0)    return -2;
 96:                             /* 見つかったファイルをオープン */
 97:     if(load) RMResLoad();   /* あらかじめメモリにロードする場合 */
 98:     RMCurResSet(curRes);    /* カレントリソースを元に戻す */
 99:     return 0;
100: }
101: /*
102:      (2) リソースの獲得
103: */
104: GetMyRes(handle *myRes, long type, int id, handle *Rsc)
105: {
106:     handle  curRes;
107:
108:     curRes = RMCurResGet(); /* カレントリソースを退避 */
109:     RMCurResSet(*myRes);    /* 自分のリソースファイルをカレントに */
110:     if( (*Rsc = RMRscGet(type, id)) <(handle)0)    return -1;
111:                             /* リソースを得る */
112:     RMCurResSet(curRes);    /* カレントリソースを元に戻す */
113:     return 0;
114: }
115: /*
116:      (3) リソースファイルの廃棄
117: */
118: CloseMyRes(handle *myRes)
119: {
120:     handle  curRes;
121:
122:     curRes = RMCurResGet(); /* カレントリソースを退避 */
123:     RMCurResSet(*myRes);    /* 自分のリソースファイルをカレントに */
124: #if 1
125:     RMResRemove();          /* リソースファイルを廃棄 */
126: #else
127:     RMResClose(myResFileNam);/* リソースファイルを廃棄(変更あり) */
128: #endif
129:     RMCurResSet(curRes);    /* カレントリソースを元に戻す */
130:     return 0;
131: }
132: /*
133:     ここがメインプログラム
134: */
135: main()
136: {
137:     if( SX_init()==FALSE ){
138:         DMError(0x101,"ウィンドウがオープンできません");
139:         exit();
140:     }
141:     while( 1 ){
142:         TSEventAvail(EVENTMASK,(tsevent*)&eventRec);
143:         switch( eventRec.eWhat ){
144:         case E_IDLE:      procIDLE();     break;
145:         case E_MSLDOWN:   procMSLDOWN(); break;
146:         case E_MSLUP:     procMSLUP();    break;
147:         case E_MSRDOWN:   procMSRDOWN(); break;
148:         case E_MSRUP:     procMSRUP();    break;
149:         case E_KEYDOWN:   procKEYDOWN(); break;
150:         case E_KEYUP:     procKEYUP();    break;
151:         case E_UPDATE:    procUPDATE();  break;
152:         case E_ACTIVATE: procACTIVATE();break;
153:         case E_SYSTEM1:   procSYSTEM();  break;
154:         case E_SYSTEM2:   procSYSTEM();  break;
155:         case E_USER1:     procUSER();     break;
156:         case E_USER2:     procUSER();     break;
157:         }
158:     }
159: }
160: /*
161:     もろもろの初期化
162: */
163: SX_init()
164: {
165:     task    taskBuf;
166:
167:     TSGetTdb(&taskBuf, -1);
168:     if( (TSTakeParam(&taskBuf.command,&winSize,NULL,0,NULL,NULL)&1)==0 ){
169:         *(int *)&winSize.left = TSGetWindowPos();
170:         winSize.right  = winSize.left + WINWIDTH;
171:         winSize.bottom = winSize.top  + WINHIGHT;
172:     }
173:     winPtr=WMOpen(NULL,&winSize,(LASCII*)WINTITLE,TRUE,WINDEFID,
174:                   (window *)-1,TRUE,TSGetID());
175:     if( winPtr == NULL ) return( FALSE );
176:     winPtr->wOption = WINOPT;
177:     activeFlag=FALSE;
178:     ctrlFlag  = CtrlPrepare();
179:     menuFlag  = MenuPrepare();
180:     CalcCardXY();
181:     ResFileName(myResFileNam);/* リソースファイル名を作る */
182:     resFlag=OpenMyRes(&myResFileHdl,myResFileNam,0);
183:                              /* リソースマップをメモリに読み込む */
184:     srand(time(0));          /* 乱数の初期化                       */
185:     InitCards();             /* カードを初期化する                 */
186:     cardopen=-1;             /* まだ一度もめくってないことに      */
187:     drawGrowBox();
188:     lastWhen=-1;             /* ダブルクリック用 */
189:     return( TRUE );
190: }
191: /*
192:     終了処理
193: */
194: SX_term()
195: {
196:     if( ctrlFlag ) CtrlDispose();
197:     if( menuFlag ) MenuDispose();
198:     if( resFlag )  CloseMyRes(&myResFileHdl);
199:     WMDispose( winPtr );
200:     exit();
201: }
202: /*
203:     グローボックスを描く(だけ)
204: */
205: drawGrowBox()
206: {
207:     GMSetGraph( &winPtr->wGraph );
208:     WMDrawGBox( winPtr );
209: }
210: /*
211:     コントロールの初期化(いまはしない)
212: */
213: CtrlPrepare()
214: {
215:     return( FALSE );
216: }
217: /*
218:     メニューは2つの項目だけ
219: */
220: MenuPrepare()
221: {
222:     menuHdl=MNConvert(/*menuHdl*/0,
223:                       /*ITEM*/"初期化する,終了する",
224:                       /*MDEFID*/0);
225:     return( (menuHdl<(menu**)0)? FALSE : TRUE);
226: }
227: /*
228:     コントロールの廃棄(いまはない)
229: */
230: CtrlDispose()
231: {
232:     return( FALSE );
233: }
234: /*
235:     メニューの廃棄
236: */
237: MenuDispose()
238: {
239:     MMHdlDispose( menuHdl );
240:     return( TRUE );
241: }
242:
243: procIDLE()
244: {
245:     return( FALSE );
246: }
247:
248: procMSLDOWN()
249: {
250:     if( (window*)eventRec.eWhom != winPtr ) return( FALSE );
251:     if( activeFlag == FALSE ){
252:         WMSelect( winPtr );
253:         activeFlag = TRUE;
254:         if( EMLStill() == 0) goto checkDClick;
255:     }
256:     switch( SXCallWindM(winPtr,(tsevent*)&eventRec) ){
257:     case W_INCLOSE:
258:         SX_term(); break;
259:     case W_INGROW:
260:     case W_INZMOUT:
261:     case W_INZMIN:
262:         GMClipRect(&winPtr->wGraph.grRect);
263:         CalcCardXY();/* 大きさが変わったらカードの縦横を再計算して */
264:         WIPE();       /* 画面を消去してアップデートイベントに備える */
265:         break;
266:     }
267: checkDClick:
268:     if(lastWhen==(-1))
269:         lastWhen=eventRec.eWhen;
270:     else{
271:         if((eventRec.eWhen-lastWhen)<EMDClickGet()){
272:                   /* ダブルクリックでカードの表裏を反転 */
273:             point_t pt;
274:             int     ix,iy;
275:
276:             lastWhen=-1;
277:             GMSetGraph( &winPtr->wGraph );
278:             pt.x_y=EMMSLoc();/* マウス座標をカード座標に変換 */
279:             if( (pt.p.x > cardxy0.p.x) && (pt.p.x < cardxy1.p.x) &&
280:                 (pt.p.y > cardxy0.p.y) && (pt.p.y < cardxy1.p.y) ){
281:                  ix=(pt.p.x-4)/cardx;
282:                  iy=(pt.p.y-4)/cardy;
283:                  if(CARD[iy][ix]<0){        /* カードが裏だったら */
284:                      CARD[iy][ix] &= 0x7f;/* 表にする */
285:                      if(cardopen){
286:                          if(cardopen<0){   /* 初めてめくるとき */
287:                              cardEven_x = ix;
288:                              cardEven_y = iy;
289:                              cardEven_id= CARD[iy][ix];
```


▶11月6日～30日までの間2,000円で過ごした。確かに図書券は数枚あった。確かに少々の食料の買い溜めはあった。だが……皆さんこんな真似をしてはいけません。ひもじくて枕を濡らした日もあった。こんな私は医学生。　　　岡本　壮紀(21)愛知県

```
290:                                 cardopen=1;  /* 一枚目を覚える */
291:                             }
292:                             else{                /* 二枚目をめくったとき */
293:                                 cardOdd_x = ix;
294:                                 cardOdd_y = iy;
295:                                 cardOdd_id= CARD[iy][ix];
296:                                 if((cardEven_id%13)==(cardOdd_id%13))
297:                                     cardopen=-1;  /* 番号が一致 */
298:                                 else
299:                                     cardopen=0;    /* 番号が不一致 */
300:                             }
301:                         }
302:                         else{             /* 一枚目をめくったとき */
303:                             PutCard(0,cardEven_x,cardEven_y);
304:                             PutCard(0,cardOdd_x ,cardOdd_y);
305:                             CARD[cardEven_y][cardEven_x]|=0x80;
306:                             CARD[cardOdd_y ][cardOdd_x ]|=0x80;
307:                                           /* 既にめくったカードを裏にする */
308:                             cardEven_x = ix;
309:                             cardEven_y = iy;
310:                             cardEven_id=CARD[iy][ix];
311:                             cardopen=1;  /* 一枚目を覚える */
312:                         }
313:                         PutCard(CARD[iy][ix],ix,iy);
314:                                           /* いまめくったカードを表示する */
315:                     }
316:                 }
317:             }
318:             else
319:                 lastWhen=eventRec.eWhen;
320:     }
321:     TSGetEvent(EVENTMASK,(tsevent*)&eventRec);
322:     return( TRUE );
323: }
324:
325: procMSLUP()
326: {
327:     return( FALSE );
328: }
329:
330: procMSRDOWN()
331: {
332:     int item;
333:
334:     if( (window*)eventRec.eWhom != winPtr ) return( FALSE );
335:     GMSetGraph((graph*) winPtr );
336:     item = MNSelect( menuHdl, eventRec.eWhere );
337:     TSGetEvent( EVENTMASK, (tsevent*)&eventRec );
338:     if(item==1){        /* 初期化する */
339:         WIPE();         /* 画面を消去して            */
340:         InitCards();    /* カードをシャッフルして */
341:         DRAW();         /* 再描画する                 */
342:         drawGrowBox();
343:     }
344:     if(item==2) SX_term(); /* 終了する */
345:     return( TRUE );
346: }
347:
348: procMSRUP()
349: {
350:     return( FALSE );
351: }
352:
353: procKEYDOWN()
354: {
355:     return( FALSE );
356: }
357:
358: procKEYUP()
359: {
360:     return( FALSE );
361: }
362:
363: procUPDATE()
364: {
365:     if( (window*)eventRec.eWhom != winPtr ) return( FALSE );
366:     WMUpdate( winPtr );
367:     if( ctrlFlag ) CMDraw( winPtr );
368:     WMUpdtOver( winPtr );
369:     DRAW();
370:     drawGrowBox();
371:     TSGetEvent(EVENTMASK,(tsevent*)&eventRec);
372: }
373:
374: procACTIVATE()
375: {
376:     if( (window*)eventRec.eWhom == winPtr )    activeFlag = TRUE;
377:     else if( eventRec.eWhom != NULL ){
378:         if( activeFlag ) {
379:             activeFlag = FALSE;
380:             TSGetEvent(EVENTMASK,(tsevent*)&eventRec);
381:         }
382:     }
383:     return( TRUE );
384: }
385:
386: procSYSTEM()
387: {
388:     switch( ((tsevent*)&eventRec)->what2 ){
389:     case CLOSEALL:
390:     case ENDTSK:
391:         SX_term(); break;
392:     case WINDOWSELECT:
393:         WMSelect( winPtr ); break;
394:     }
395: }
396:
397: procUSER()
398: {
399:     return( FALSE );
400: }
401: /*
402:     指定した領域にカードを拡大／縮小しながら表示する
403: */
404: PutImg(rimg,drect)
405: short    *rimg;  /* rectImg型のデータ(PAT4) */
406: rect     *drect; /* 表示する領域 */
407: {
408:     bitmap  bm;
409:     rect    srect;
410:
411:     bm.bmKind        = G_TXT;
412:     bm.bmRect.left   = rimg[0];
413:     bm.bmRect.top    = rimg[1];
414:     bm.bmRect.right  = rimg[2];
415:     bm.bmRect.bottom= rimg[3];
416:     bm.base          = (int)(rimg+4);
417:     bm.line          = (rimg[2]-rimg[0]+1)>>3;
418:     bm.opt.tbm.page = bm.line*(rimg[3]-rimg[1]);
419:     bm.opt.tbm.aPage= 15;
420:     srect            = bm.bmRect;
421:
422:     GMSetGraph(&winPtr->wGraph);
423:     GMTransImg(&bm,winPtr->wGraph.bmap,&srect,drect);
424: }
425: /*
426:     ５２枚のカードを描画する
427:
428:     CARD[x][y]の値 bit7:    1なら裏, 0なら表
429:                    bit7-0: カードの番号(1～52)
430: */
431: DRAW()
432: {
433:     int     ix,iy;
434:
435:     GMSetGraph( &winPtr->wGraph );
436:     GMAPage( 15 );
437:     for(iy=0;iy<4;iy++)
438:         for(ix=0;ix<13;ix++){
439:             PutCard(( CARD[iy][ix]<0) ? 0 : CARD[iy][ix], ix,iy);
440:         }
441: }
442: /*
443:     画面を消去する
444: */
445: WIPE()
446: {
447:     int     fc;
448:
449:     fc = GMForeColor( G_LGRAY );
450:     GMFillRect( &(winPtr->wGraph.grRect) );
451:     GMForeColor( fc );
452: }
453: /*
454:     カードを置くときの縦横のドット数を再計算する
455: */
456: CalcCardXY()
457: {
458:     int left,top,right,bttm;
459:
460:     left  = winPtr->wGraph.grRect.left   +4;
461:     top   = winPtr->wGraph.grRect.top    +4;
462:     right = winPtr->wGraph.grRect.right -20;
463:     bttm  = winPtr->wGraph.grRect.bottom-20;
464:     cardxy0.p.x = left;
465:     cardxy0.p.y = top;
466:     cardxy1.p.x = right;
467:     cardxy1.p.y = bttm;
468:     cardx       = (right-left)/13;
469:     cardy       = (bttm -top )/4;
470: }
471: /*
472:     カードの座標からイメージを置く
473:     レクタングルを計算する
474: */
475: rect CardRect(ix,iy)
476: int ix;
477: int iy;
478: {
479:     rect    r;
480:
481:     r.left   = cardxy0.p.x+ix*cardx+2;
482:     r.top    = cardxy0.p.y+iy*cardy+2;
483:     r.right  = cardxy0.p.x+(ix+1)*cardx-2;
484:     r.bottom = cardxy0.p.y+(iy+1)*cardy-2;
485:     return r;
486: }
487: /*
488:     カードを座標位置に表示する
489: */
490: PutCard(resid,ix,iy)
491: int resid;
492: int ix; /* 0..12 */
493: int iy; /* 0..3  */
494: {
495:     handle  card;
496:     rectImg *cardp;
497:     rect    drect;
498:
499:     if(resFlag==0){
500:         if(GetMyRes(&myResFileHdl,'CARD',resid,&card)<0){
501:             DMError(1,"リソースが見つかりません");
502:             SX_term();
503:         }
504:         cardp= *(rectImg**)card;
505:     drect=CardRect(ix,iy);
506:         PutImg(cardp,&drect);
507:     }
508:     else{
509:         DMError(1,"リソースが獲得できていませんよ");
510:         SX_term();
511:     }
512:
513: }
514: /*
515:     カードをシャッフルする
516:
517:     簡単のために CARD を1次元配列とみなして処理する
518: */
519: ShuffleCards()
520: {
521:     char    *cp=(char*)CARD;
522:     int     i,a,b,c;
523:
524:     for(i=0;i<100;i++){
525:         a=((rand()>>4)&0xfff)%52;
526:         b=((rand()>>4)&0xfff)%52;
527:         c=cp[a];
528:         cp[a]=cp[b];
529:         cp[b]=c;
530:     }
531: }
532: /*
533:     カードを初期化する
534: */
535: InitCards()
536: {
537:     int ix,iy;
538:     int id=1;
539:
540:     for(iy=0;iy<4;iy++)
541:         for(ix=0;ix<13;ix++,id++)
542:             CARD[iy][ix]=0x80|id;      /* ビット7が1なら裏を意味 */
543:     ShuffleCards();                    /* 順番に入れた後シャッフル */
544:     cardopen=-1;                       /* カードの開いた回数の偶奇 */
545: }
```

▶私は暇を見つけては，昔のMZ-80Kの本を開き，そこにあるゲームをX68000に移植して遊んだりしています。昔のゲームって技術はともかく，ゲームに愛が感じられていいですね。
今井　彰彦(27)大阪府

CGAマガジンの積極的な使い方(その1)

プロジェクトチームDōGA
かまた ゆたか

すでに発行されている(はずの)CGAマガジン創刊号。その内容を紹介するとともに，収められているデータを使って，少し本格的なCGAを制作してみましょう。

はじめに

かまた「ただいまー。トルコから帰ってきたぞ」
竹内「あっ，かまたさん，お帰りなさい。お元気でしたか？」
かまた「いやあ，最後にイスタンブールで風邪ひいて，気分悪いわ」
松井「変なインフルエンザでも拾ってきたんじゃないでしょうね。うつさないでくださいよ」
かまた「そうやったらOh!Xで告知せなあかんなあ。“DōGA，ウイルスに感染す”って」
松井「シャレになってませんよ……」

さて，今回はCGAマガジンの紹介と使い方ですが，誌面上で宣伝して，バンバン売りつけようというものではありません。このCGAマガジンは基本的にコピーフリーですから，よけいなお金を使ったり，私たちに配布の手間をかけたりせずに，サークル内や友人にコピーしてもらってください。我々はより多くの方々に楽しんでもらうことを希望します。

その際，ただ収められているアニメーションを作ってみるだけというような受け身的な楽しみ方ではなく，データを活用して，自分でいろんなCGAを作ってみるという積極的な使い方をしてください。この連載ではその積極的な使い方について，具体的に詳しく解説していきたいと思います。このCGAマガジンをきっかけに，ひとりでも多くのCGA作家が生まれることを期待します。

まず，CGAマガジンの紹介から始めますが，困ったことに，この原稿を書いている時点ではまだCGAマガジンは影も形もありません。ですから，今回の原稿は予想で書いている部分が多分にあります。実際との食い違いにつきましては，あらかじめご了承ください。

編集長はMAX田口君です。そういえば，田口君の姿をここ2，3日見ないけど大丈夫なのかなあ。2，3日前からプロジェクトルームにあったスーパーファミコンも見当たらないけど，本当に大丈夫かなあ。

CGAマガジン発行の主旨

CGAマガジンにはたくさんの形状データやフレームソースが収められており，バッチファイルによっていろんなアニメーションを自動作成することができます。当チームのスタッフ(CGAマガジン編集部)が作ったデータもありますが，基本的にはユーザーの皆さんからの投稿を中心に編集しています。

つまり，“手軽でパーソナルな映像表現としてのCGAの普及”という当チームの活動の一環として，

1) 小作品や形状データなどの発表の場を設ける
2) データベースを構築することで，作品制作の作業を軽減する

という2つの目的のためにCGAマガジンは発行されます。当面は年4回発行が目標です。

よいデータとは

今回のCGAマガジンに収められているデータを見て，“ゲ，ゲー！　こんなん人間のすることちゃうわ。こいつらみんな×××(ピー)や！”と思う方も多いでしょう。なんてったってあの面数ですからねえ。

同時に，“いきなりこんなレベルで創刊されたら，とても投稿できない”とも思われるでしょう。でも，それは違います。はっきりいって，今回の創刊号のデータは決して“よいデータ”とはいえません。

“よいデータ”とは，多くのユーザーに利用されるデータです。つまり，みんながよく使うようなものであり，そして面数も少ないということです。どんなに精密にできていても，面数が異様に多く，複数のマッピングがされていて，メモリを増設しないと作画できないとか，複数並べると作画時間がかかりすぎるといったデータは，あまり“よいデータ”とはいえません。

もちろん“よいデータ”でないから“悪いデータ”だといっているわけではありません。今回のようなデータは“すごいデータ”なのです。“ここまで凝ったデータを作ったぞ。どうだ，すごいだろう！”的なデータも，ぜひ応募してください。CGAマガジンは，この“よいデータ”と“すごいデータ”を適当に織り交ぜて発行していきたいと思います。

今後の特集としては“街”“部屋の中のもの”“BGMAKE用背景画像集”などが候補に挙がっています。特によい題材が思い当たらない方は，これにご協力ください(その場合，1cga＝1mmとか，5mmとかの合わせやすい縮尺にしておいてください)。“少ない面で最大の効果”，これがよいデータの極意です。

また，価格は“無料ではあるが無償ではない”という方針でやっていくつもりです。“無料”ですので，特に定価は存在しません。ただ，タケルで入手すればタケル使用料1,000円をブラザー工業に，ネットで入手すればネット使用料をネットやNTTに対して支払わなければいけません。友人からコピーを受け取ればタダです。しかし，ただもらいっぱなしというのはなしです。各ユーザーはCGAマガジンによって受けた恩恵や楽しみや感動の対価を，なんらかの方法で支払わないといけません。これが“無償ではない”ということです。

ルール１　腕がある人はデータを出す。それは，次のCGAマガジンとなる。

ルール２　お金がある人はカンパを出す。それは，CGAマガジン編集部の運営費となる。

ルール３　地元（関西）の人は労働力を出す。それは，CGAマガジン編集部員となる。

DōGAプロジェクトの基本理念のひとつは“各自ができる範囲で努力する”です。この３つのルールに従って，各自ができる範囲で努力すれば，正のフィードバックとなり，CGAマガジンは末永く発行を重ねることができ，アマチュアCGA界には膨大なデータが共有されるはずです。ほんまかいな？

が，本当かどうかは，皆さんが決めるのです。コピーなりなんなりしてCGAマガジンを受け取った瞬間から，自分は準CGAマガジン編集部員なんだという自覚をもって，自分のなすべきことを考えてください。

また，もうひとつのルールとして，CGA作品制作においてCGAマガジンのデータを流用した場合は，そのデータの作者の労をねぎらう意味でもエンディングクレジットなどで作者名を表示するようにしましょう。そうすることによって，たとえ作品を制作できない人でもいいデータをCGAマガジンに提供することで，たくさんの作品に名前を出すことになります。

CGAマガジンのおいしい中身

今回のCGAマガジンにはデータ以外にも，DōGA CGAシステムver.2.50の発表以降に開発されたツール，バージョンアップされたツール，またX68000以外の機種のツールなども入っています。

新しいツールが２つ，バージョンアップしたものが８つ，他機種のツールが12もあるのは活動が活発な証拠といえるでしょう（でも，田口君は全部は入りきらないから，いくつか削除するといってました）。新ツールの紹介，バージョンアップの内容につきましては，コラムにまとめましたのでそちらをご覧ください。

また，“ほかの機種のツールなんて，X68000しか持っていない私には関係ない”と思われるかもしれませんが，これは他機種ユーザーもCGA制作に引き込んじゃえということです。

特に486マシンなどをレンダリングに利用すると効率がぜんぜん違います。FFEなどのモーションデザインツールはありませんが，フレームソースの文法さえ理解すれば，エディタで動きを作り，FFでフレームファイルにしてレンダリング，アニメーションを見て確認，という一連の作業ができます。データフォーマットはX68000とまったく同じです。作画した画像データをX68000にかければ，そのままアニメーション，録画できますから，作品制作の分業も十分可能でしょう。

音楽のデータも入っています（予定）。CGA作品制作においては，BGMの著作権がいつも問題になります。ですから，著作権上問題のない曲を集めてCGA作家の皆さん

バージョンアップ一覧

［新ツール］

○TCHED（タイムチャートエディタ）

本格的に長い作品を作るときは，その編集作業が結構大変でした。HANIMで画像を読み込み，アニメーションさせて，タイムチャートを修正して，またHANIMで……。特に画像読み込みで非常に時間がかかります。

TCHEDはアニメーションの最中にコプロセスでエディタに入り，タイムチャートを修正できるツールです。画面サイズを極端に小さくすることで，一度にアニメーションできる量もHANIMの数倍になりますので，とりあえず使いそうな画像を全部メモリに入れて，長編を編集することができます。

○MOB（モーションブロー）

Ｉ回生が練習用に作った簡単なツールです。動画の前後の画像を合成することで，モーションブロー（動きの速い物体が流れるような効果）を得ます。先月号のCGAマガジン告知のサンプル画像をご覧ください。来月号で，実際に使用してみるつもりです。

［PC-9801関連ツール］

○REND，FF，SLIDE，MKTCH，SRANIM，STAR，PILEほか

386，486マシン対応版もあり，作画の分業などが可能になります。

［バージョンアップツール］

○EPA2

強力なマクロ機能がつきました。たとえば，“全体をぼやけさせて，赤い部分を光らせる”という処理をマクロとして登録すると，動画データに対して，１枚読み込んではマクロ処理をして，セーブして，また次を読み込んで……，ということを実行します。つまり，動画対応のペイントソフトとなったのです。

作者である宇宙人森山氏は，この機能を生かしたCGA小作品を制作して，CGAコンテストに応募するとのウワサです。

○BGMAKE

画角に対応しました。実は，前のバージョンでは視点を望遠にしても広角にしても，出力される画像は約45度分という，ほとんどバグに近いような仕様がありました。これで，3Dで作画する物体の背景と合成しても違和感がほとんどなくなります。

それから，ヘルプメッセージが日本語になりました。前のバージョン制作の際に，ヘルプは日本語で統一しようといっていたのに，作者のＰ君がポリシーとして英語がいいと突っ張ったのです。でも，最近できた彼女が「日本語のほうがわかりやすい」とひとこといったら……。

○FF

乱数を発生させる関数が加わりました。うまく使えば，FIの路面による振動なども表現できます。これも次号で挑戦してみましょう。

○IC，KAMA，SCROL，BETA

バグが減ったり，精度が上がったりしたそうです。

に提供していこうというわけです。作曲やクラッシックのアレンジなどができる方は，ぜひともデータの提供をお願いいたします。システムとしては，Z-MUSICを利用させていただきます。

そのほか，データ提供者の方の生の声なども載せていますので，お楽しみに（電脳倶楽部みたい……）。

CGAマガジンの基礎的な使い方

すでに解説したように，このCGAマガジンには各種アニメーションのデータとバッチファイルが入っていますので，メニューから好きなバッチファイルを選択してもらえば，あとは自動的にアニメーションを制作します。

メニューは，1992年7月号付録ディスクでの「お試しシステム」のようなものを制作する予定でしたが，開発が遅れ，「満開製作所」のご好意により「D_SHELL」を使用させていただきます。これで“電源オンですぐ起動。マウスひとつでらくらく操作”です。

TAKA2「かまたさん，“D_SHELL”だとサンプル画像がモノクロになってしまいますよ」

かまた「へ？ そうなん。でも，時間がないからええわ」

TAKA2「実は，以前に“Ko-WINDOW”上で動く“D_SHELL”コンパチソフトを作ったんですけど，あれを改造したらカラーになりますよ」

かまた「それやったら，そっちでええやん。でも，なんでそんなソフトを作ったん？」

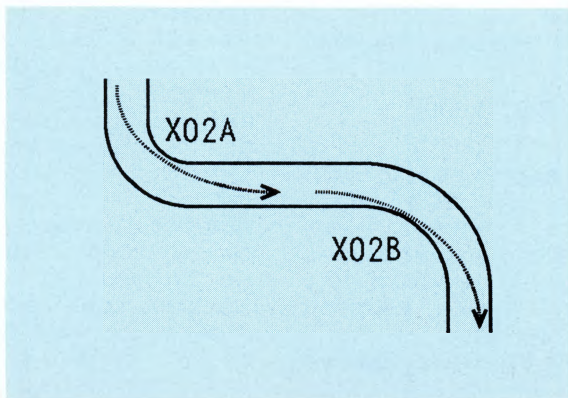
TAKA2「“Ko-WINDOW”上で電脳倶楽部が読めるじゃないですか」

かまた「意味がなーい！」

というわけで満開製作所の祝さん，お騒がせしてもうしわけありません。

CGAマガジンの積極的な使い方

基礎的な使い方が用意されたアニメーションをレンダリングしてみるだけなのに対して，積極的な使い方とは形状データを利用し，自分で動きやカメラワークを設定

図1　シケインのカット

して，オリジナルのカットを制作することです。それでは実際に1カット制作してみましょう。

○今回作るカットの概要

シケインを走り抜けるカット（図1）を作ってみます。以前述べたように，1カットで左右に曲がるモーションを作るのは難しいので，左に曲がる前半を「X02A.FSC」，右に曲がる後半を「X02B.FSC」と，2つのカットに分けて制作します。

まず今回は，道路と車のデータだけでモーションデザインをし，次回いろんな小道具を使って仕上げます。今回は背景がほとんどないのでスピード感がありませんが，次回ちゃんと仕上げればかっこよくなる……はずです（あまり自信がない）。

○データの展開

CGAマガジンに収められている形状データなどは，ディスク容量の問題からすべて圧縮されています。必要なデータを展開するところから始めましょう。

まず，CGAマガジンを起動します。ドライブ0にCGAマガジンを入れて，「電源ON！」。ハードディスクからの起動になっている場合は，「OPT.1」キーを押しながら電源を入れてください。Ko-WINDOWが立ち上がり，ウィンドウが開いて，その中に“Ko-SHELL”（Ko-SHELLとは，「Ko-WINDOWのD_SHELL」の略）によるメニューが表示されます。

電脳倶楽部を購読している方は予想がつくでしょうが，“Ko-SHELL”では，マウスの左ボタンで下へ，右ボタンで上へ画面をスクロールさせることができます。メニューの左端のボタンをクリックすると，選択，実行し，上のメニューに戻るときは，左右同時クリックです。

メインメニュー（もくじ）の下のほうに「今回のデータベース」があります。ここを選択すると，内容別のメニューが表示されますので，「特集：走れ！ F1のデータ」を選択します。すると，この特集で使われている形状データの一覧が表示されます。

今月使うのは「F1の4車種」と「道路各種」の2種類です。左端のボタンをクリックすると，まずサンプル画像と各データの詳しい説明が表示されます。

またここで，データをフロッピーディスクに展開するか，ハードディスクに展開するかを指定することができます。ハードディスクをお持ちの方は，あらかじめハードディスクにCGA制作用のディレクトリ（例 ¥CGAWORK）を作っておき，そこにデータを展開すれば，以後の作業も楽になります（この場合，ハードディスクはCドライブになります）。フロッピーディスクに展開する場合は，フォーマットずみの十分な空き容量のあるディスクが必要ですが，フォーマットをする機能もありますのでご利用ください。

「F1の4車種」を展開すると，

WILLI.SUF，WILLI.ATR，SWILL.SUF

BENET.SUF, BENET.ATR, SBENE.SUF
JORDA.SUF, JORDA.ATR, SJORD.SUF
TYRRE.SUF, TYRRE.ATR, STYRR.SUF
という12個のファイルを作ります。

頭に「S」がついているのはシンプル版です。シンプル版というのは，FFEでモーションデザインするときに使用する形状データです。モーションデザインのときは何度も表示を繰り返しますが，通常の形状データでは面数が多く，待たされてイライラします。そんなに細かくなくても，だいたいのイメージがわかれば十分なのですから，面数を大幅に減らしたデータを代わりに使うというわけです。RENDで作画するときは，モーションデータ（フレームソース）はそのまま使用し，形状ファイルはちゃんとした形状データのほうを指定します。

ウイリアムズ，ベネトン，ジョーダンは私が半日かけて作った比較的簡単なデータです。ティレルは，“お試しシステム”のサンプルデータに大きさなどの修正を加えたものです。

「道路各種」を展開すると，

ROAD.ATR　：全道路共通のアトリビュート
ROADMAP.PIC：道路のマッピングデータ
STRAI.SUF　：直線（20m）
M_STR.SUF　：直線のマッピング版
EDGE.SUF　：縁石
R10.SUF, R10H.SUF, M_R10.SUF, M_R10H.SUF
R30.SUF, R30H.SUF, M_R30.SUF, M_R30H.SUF
R60.SUF, R60H.SUF, M_R60.SUF, M_R60H.SUF
R100.SUF, R100H.SUF, M_R100.SUF, M_R100H.SUF
：それぞれ半径10，30，60，100mのコーナー
という21個のファイルを作ります。

道路のアトリビュートはどのパーツも同じなので，1つしかありません。形状ファイルで頭に「M_」がついているのはマッピング対応版で，タイヤの跡のついた「ROADMAP.PIC」が張り付きます。メモリや作画時間に余裕がある方はそちらを使うとよいでしょう。「_」がファイルネームにつくと，CADで読み書きできないのですが，もともとCADはマッピングに対応していないので，わざとこういう名にしました。

カーブの形状データについてはあとで詳しく説明しますが，「R＊.SUF」は30度分，「R＊H.SUF」は15度分曲がります。ですから，「R＊.SUF」を3つつなげると90度曲がり，「R＊.SUF」と「R＊H.SUF」を1つずつつなげると45度曲がるわけです。

○制作に入る前に

今回制作するカットは，道路のパーツを並べてコースを作るところから始めます。しかし，このようにコースを作る場合に，資料を取り出していきなり鈴鹿サーキットの全体を作るようなことは絶対にしないでください。コースは道路だけでなく，ガードレールやスポンサーの看板，芝生，樹木なども含まれますから，鈴鹿サーキット全体のデータ量は膨大で，メモリが何メガあっても作画できるものではありません。S字コーナーでのドッグファイトのカットではS字だけ，ヘアピンからホームストレートの立ち上がりのカットではヘアピンとホームストレートだけを作ります。F1のTV中継などの映像を見ても，全コースが写っているカットなんてほとんどないでしょう。

パーツを並べる際に注意しなければいけないことはいくつかありますが，まず縮尺について解説します。CGAマガジンでは，できるだけ縮尺を統一する方針です。少なくとも私が担当した「走れ！ F1」では，2種類に統一されています。

まず，今回展開した形状データ「F1の4車種」「道路各種」などは1cga（コラム「CGAの単位」参照）が1cmに相当します。そして，「藤井マクラーレン」や「古本フェラーリ」など特に細かく作られている物体は，1cgaが2mmになっています。ですから，「道路」の上にそのまま「藤井マクラーレン」を置くと，異様に大きく，道路からあふれてしまいます。この場合，縮尺が5倍違うのですから，「藤井マクラーレン」を置くときに，X，Y，Z軸方向に0.2倍ずつ縮少するか，道路のほうを5倍ずつ拡大しておく必要があります。今月の制作では「藤井マクラーレン」も「古本フェラーリ」も使いませんから，1cga＝1cmとして考えてください。

さて，今回のシステム設定としては，以下のようになっているものとして話を進めます。

CGAの単位

比較的に初心者の方は，“CADやFFEで使用する数値の単位はなんだろう”と疑問に思われるようです。しかし，“mかcm，もしくはmm。まさかkmということはないだろう”などと考えても無駄です。

実はCGAには長さの単位はないのです。なぜなら，そもそも単位とは相対的なもので……と長々と説明しているといよいよ混乱するばかりでしょうから（説明が悪いだけ），ここで大ボラを吹くことにします。

CGAで使用する長さの単位は“cga”なのです。ただ，“kcga（キロcga）”とか“mcga（ミリcga）”とかはなく，“cga”の1種類だけですから，“1000cga”を省略して“1000”と表記することが多いのです。年齢を聞かれて，“22です”と答えれば，“22歳”といわなくてもわかるでしょう。それとまったく同じです。

ほ〜ら，なんとなくわかったような気になったでしょう（ならへんって？）。

しかし，決してCGAの専門家の方に，“この物体の全長はおよそ2500cgaです”なんていわないでくださいね。

“なんですか，それ？”と聞かれるのがオチですから。

○メインメモリ2Mバイト
○コプロセッサなし
○作業用ディレクトリはハードディスクのA:¥CGAWORK
○CGAシステムはハードディスクにインストールずみ
○CGAマガジンについてきたバージョンアップツールなどもインストールずみ（今回は新バージョンのFFEを使用します）
○A:¥DOGACGA にパスが設定されている

メモリが1Mバイトのままの方やハードディスクをお持ちでない方もいらっしゃるでしょうが，やはり本格的にCGAをするためには必需品といえます。ただ，今回制作するカットはこれだけの環境がないとできないというわけでなく，作業手順などが若干異なるということで，絶対に不可能というわけではありません。しかし，ハードディスクがないと，作画の途中でディスクを交換しなければならないなど，かなりの手間がかかることは予想されます。

○新FFEの使い方

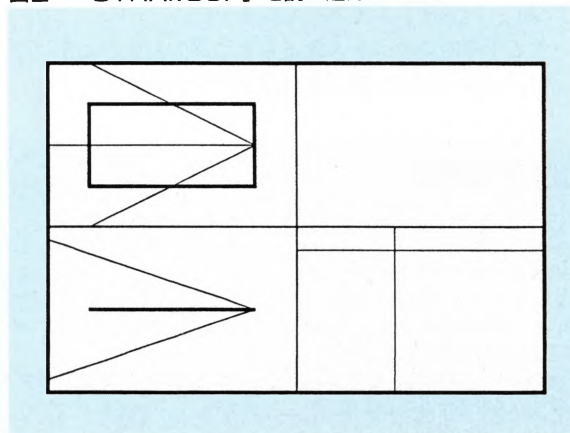
A:¥CGAWORKにカレントディレクトリを移動したら，まず「DIR」を実行して，上記の展開したデータがちゃんとあることを確かめてください。このあたりの操作がわからない方は，マニュアルの「CGA大学/教養講座/コンピュータ基礎概論」(T-27)をご覧ください。

今回もPESは使わずに，コマンドラインからの入力で解説します。そのほうが誌面では解説しやすいからです。

それでは，さっそく新バージョンのFFEを起動しましょう。従来どおり，

　　ffe ＜リターン＞

です。起動後の画面もほとんど変わりません。平面図と側面図の中央に，注目点を意味する小さな赤い四角が加わった程度です。

とりあえず，物体をひとつ置きます。マウスでメッセージパネルの「物体設定」をクリックして，さらに「追加」を選択すると，先ほど展開した物体名がずらずらと表示されます。「▼」をクリックすれば表示しきれなかった分がスクロールして出ますが，そんなことなどしなくても「STRAI.SUF」はちゃんと表示されていると思います。「STRAI.SUF」をクリックすれば，少し間をおいて，画面に道路が黄色で表示されます（図2）。

このあとメッセージパネルは入力モードとなり，位置や拡大などが表示されていますが，変更せずにそのまま「決定」をクリックします。これで「STRAI.SUF」をひとつ置くことができました。以後，このようにメニューをマウスで選択，実行していくような操作を「物体設定/追加/STRAI.SUF/決定」と省略して記述します。

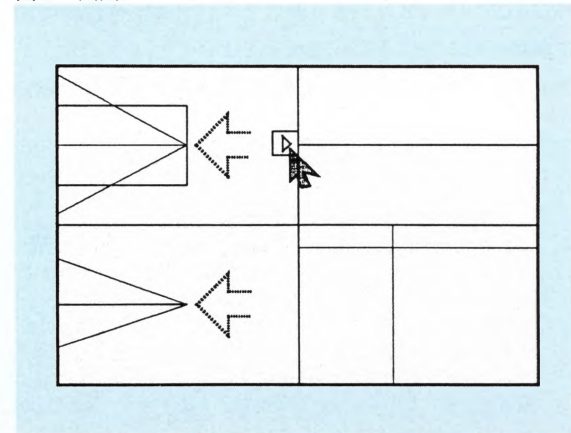
この「STRAI.SUF」はストレートのパーツです。道路幅は10m（Y座標±500cga)，長さは20m（X座標±1000cga)だけですので，長い直線コースを作る場合は複数の「STRAI.SUF」を並べるか，X座標のスケールを大きくする必要があります。

では，もうひとつ「STRAI.SUF」を並べてみましょう。まず，表示画面（画面左半分）をスクロールします。これは平面図，側面図の上下左右にある「▲」マークをクリックするのですが，スクロール方向がCADとは逆なので注意してください（CGAシステムは複数のプログラマが開発しているため，お互いにポリシーを譲らず，このように設計思想に食い違いが出ていることがよくある）。つまり，X軸の＋方向（画面右側）の領域を見たい場合には，画面右端にある「▲」をクリックするわけです（図3）。

現在はまだ物体設定モードですから，「追加/STRAI.SUF」として，もう一度「STRAI.SUF」をLOADします(黄色で重なって表示される)。しかし，その位置で決定せずに，前の「STRAI.SUF」にX軸の＋側につなぎます。座標位置は（2000,0,0）ですが，ここは数値入力ではなく，マウスで位置指定してみましょう。

従来のマウスによる位置指定は1cga単位だったため，2000ぴったりにするのはかなりの手間だったのですが，新しいFFEでは10cga単位で指定できますし，移動単位の変更もできます。「視点/画面」をクリックするとカウントという数値が表示されますので，「▼」を3回クリッ

図2　「STRAI.SUF」を読み込んだところ

図3　画面スクロール

クして100cgaにしてください。平面図で図４のようにマウスで位置を指定してクリックすると，白いマークが現れ，その座標が表示されますので，簡単に（2000,0,0）に合わせられるはずです。なお，この状態でリターンキーを押せば平面図，側面図に現在の位置が表示されますので，ちゃんとつながっていることを確認してから，「決定」してください。

物体選択も，マウスで指定できるようになりました。「変更」を選択してみてください。従来はここで選択したい物体の位置座標を入力していましたが，新バージョンではマウスでクリックして物体を指定できます。

試しに，平面図上で先ほど（2000,0,0）に置いた「STRAI.SUF」をクリックしてみてください。選択されたほうの「STRAI.SUF」の中央に小さな四角が表示されます。動かしたい物体をちゃんと選択できたのを確認して「決定」します（別の物体を選択してしまったときは「選択」で次の候補に移ります）。水色の表示が黄色に戻り，入力モードになります。選択した「STRAI.SUF」をＸ軸の－側（－2000,0,0）に置き直してみてください。この場合，画面をスクロールするより，テンキーの「＝」を押して，表示範囲を変更したほうがやりやすいでしょう（図５）。座標を指定できたら「決定」で変更モードから抜け，「終了」で物体設定から抜けます。

マウス指定が強化されたのは物体を置くときだけではありません。「視点設定」に入ってみてください。従来，視点位置や注目点位置は数値入力で行っていたのですが，これもマウスで指定できるようになりました。ただ，マウスで位置を指定するだけですと，その位置が視点なのか，注目点なのかがわかりません。そこで，数値入力用のカーソル（数値が反転表示されている部分）が視点の座標にあるときは視点の位置指定，注目点の座標にあるときは注目点の位置指定となります。

ですから，注目点の位置を指定したいときは，まず注目点の位置座標（Ｘ，Ｙ，Ｚのどれでも可）が表示されているところを一度クリックしてから，注目点の位置を平面図，側面図上で指定します。注目点を表す小さな赤い四角が移動し，同時に視野の表示も変更されます。

「視点設定」に入ったときは視点位置設定の状態になっていますので，図６のように側面図で視点の位置を指定して「作画」をさせると，透視図もちゃんと見下ろした完成予想図が表示されます。視点，注目点を適当に動かし，気に入ったところで「決定」してください。以前と比べて便利になったでしょう？

FFE以外にもバージョンアップしたツールはありますが，それについては別コラムにまとめましたので，そちらをご覧ください。

○コースを設定する

今回制作するカットのシケインを図７に示します。先ほど試しに置いた「STRAI.SUF」にほかのパーツをつけ足して，作っていきましょう。

まず，Ｘ軸の＋側のカーブから作ります。「STRAI.SUF」と同様に，「物体設定/追加/R30.SUF」で「R30.SUF」を呼び込みます。が，「STRAI.SUF」のときのように原点には現れません。なぜなら，カーブのパーツはすべてカーブの曲率の中心を原点にしているからです。

図８をご覧ください。「R30」とは半径（曲率の中心から道路の中央まで）が30m，つまり3000cgaであることを

図４　マウスのカウント量の変更

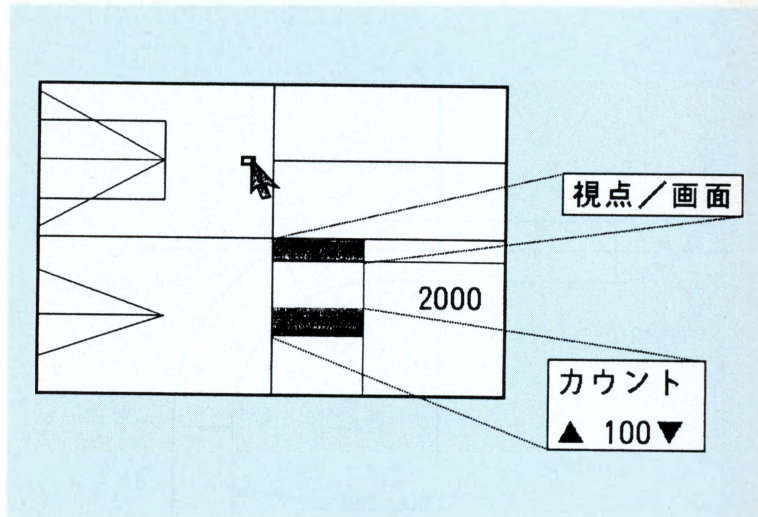



図５　「STRAI.SUF」を移動する

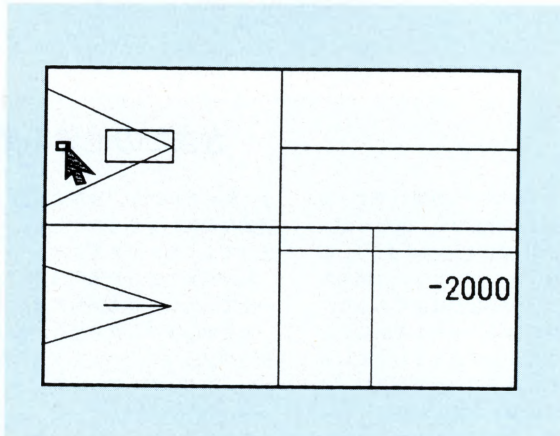
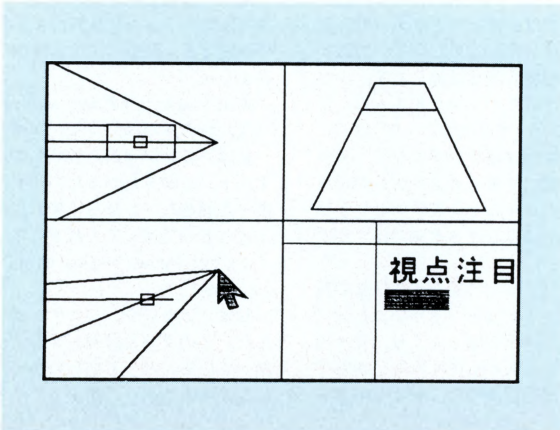
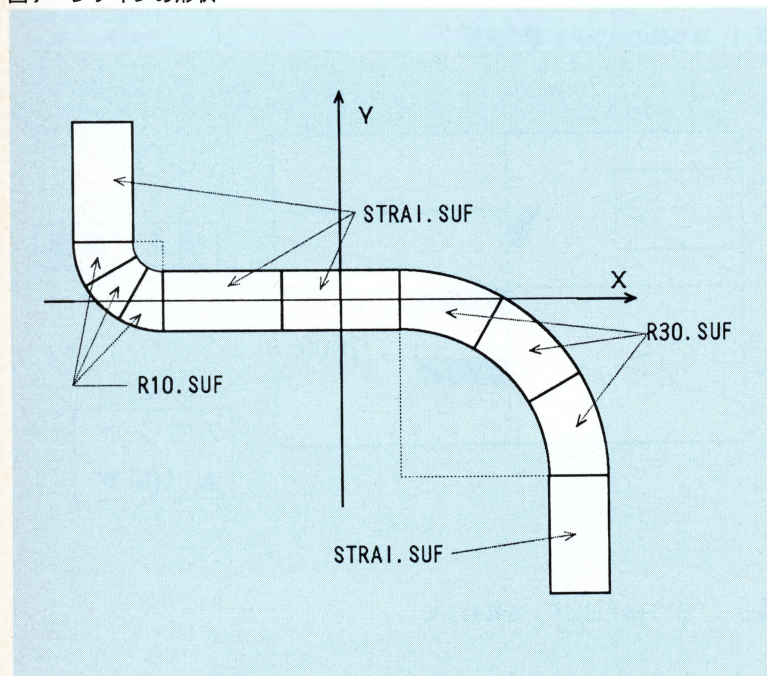



図６　視点位置設定

意味しています。原点は曲率中心にありますから，「STRAI.SUF」の横にそのまま「R30.SUF」を置いてもつながりません。半径の分だけずらす必要があるのです。「STRAI.SUF」の端の座標は（1000,0,0）ですから，「R30.SUF」の座標は（1000,−3000,0）となります。

なぜ，このように原点の位置をずらしているのでしょうか？　それは次の操作で明らかになります。「R30.SUF」は前にも述べたように30度分しかありませんから，図7のように90度曲げるためにはあと2つ「R30.SUF」をぴったりくっつけて並べなければいけません。

しかし，直線コースならともかく，30度分だけ曲がったところの座標など，通常は関数電卓でも計算しないとわかりません。ところが，この「R30.SUF」のように原点を曲率中心に置いてやると，位置座標はまったく同じで，Z軸回りの回転角度を変えるだけで，ぴったりくっつくのです(図9)。具体的には，あと2つの「R30.SUF」は位置はともに（1000,−3000,0）で，Z軸回りの回転角度がそれぞれ−30度と−60度になります。

同様に，X座標が−側のカーブも作ってみましょう。こちらのカーブは半径10mの「R10.SUF」を使用します。「STRAI.SUF」の端の座標は（−3000,0,0）です。だからといって，「R30.SUF」と同様に半径の分だけずらして（−3000,−1000,0）に置くだけでは，図10のようになってしまいます。

Z軸回りに30度回転すればいちおうくっつきますが，カーブの曲がり方が逆になってしまいます。どうしましょう……，というほどの問題ではありませんね。拡大率を−1倍すればよいのです（−1倍すると，左右，前後など反転する）。この場合はZ軸回りに回転するのもやめて，位置を（−3000,1000,0），X方向，Y方向それぞれの拡大率を−1.0にします。

このカーブも90度ありますから，あと2つ「R10.SUF」を並べます。位置は同様に（−3000,1000,0），XとY方

図7　シケインの形状

かまたの地球の歩き方

このコラムはフリーツアーへの誘いです。コンピュータにもCGAにも関係ない，単なる個人的な趣味のコラムです(実は私は旅オタク)。ほかのパソコン専門誌でも，パソコンとは関係ない映画や本の紹介コーナーなどはよくあるし，某誌には趣味のお料理のコーナーまであるのだから，Oh!Xに旅行のコラムがあってもいーじゃないか，ということに勝手に決めてしまい，そこそこ人気があれば連載します。

ひとりで海外旅行へ行こうとすると，まず心配なのが言葉の問題ですが，これはなんとかなるものです。

少なくとも語学力は必要ありません（私も英語は中学で落ちこぼれた)。まずたいていの国では，我々が話せる程度の英語は通じます。少なくとも，ホテルに行って“シングルルーム，ワンナイト”，お店に行って“ハウマッチ”はどこでも通用します。問題は語学力ではなく，自分の知っている英単語（つまりカタカナ）の組み合わせだけで相手にわからせる表現力と，多少通じなくても積極的に話かける度胸のほうがずっと大切なのです。

アメリカやイギリスなどの英語圏では私の英語は通用しません。なぜなら，そういった国では相手は英語をベラベラと話しまくり，我々の発音が悪いとちっともわかってくれないからです。それに対して非英語圏では，相手も英語をよく知らないので，片言の英語（つまりごく基礎的な単語）をたどたどしくしゃべるため，かえってよくわかりあえます。

私が初めて行った海外はアメリカのニューヨークでした。ここの街角には，必ずといっていいほどホットドッグの屋台があります。そんな屋台のひとつで，ホットドッグを頼みました。

“ホットドッグ，プリーズ”

“ハ，ハ〜ン？”

全然通じません。でも，その屋台で売っているのは，ホットドッグしかないのです。

また，ニューオリンズへ行ったときの話です。果物屋でプラムを1袋買いました。そこで果物屋のおばさん相手に，少し高度な英語を使ってみました。

“Must I wash to eat this puramus ?”

（注，私の英語はつづり，文法ともデタラメ）

私は「このプラム，洗わずに食べられるのですか？」と聞きたかったのですが，おばさんは困った顔をしてしばらく考えたあと，バナナを1本ちぎって渡してくれました。

“No, I don't need banana! Must I wash to eat this puramus ?”

私は「いやいや，バナナがほしいんじゃないんだ。このプラムは洗わずに食べられるのかと聞いているんだ」といいたかったのですが，おばさんはまた困った顔をして，今度はバナナを1房くれました。私は黙って握手をして，立ち去りました。

中国へ行ったこともあります。一般人民で日本語，英語が話せる人はあまり多くありませんでしたが，ホテルなどではあまり不自由しませんでした。

“Hello! Can I take a singleroom tonight?”（こんにちは。今晩シングルルームに泊まりたいのですが）

“Ok. Your room no.701. Your room is another billding Sei-rou.”（はい，701号室へどうぞ。あなたの部屋は別館の西楼になっております）

私はその日，シャワーだけでなく，お風呂に入りたかったので，

“There are bath?”（その部屋には，お風呂がありますか？）

と聞いてみました。すると，受付のお嬢さんは驚いて，

“No! This hotel is not so wide. You need not bus!”（このホテルは，バスがいるほど広くありません！）

＊　＊　＊

旅に出ると，いい人々とそうでない人々に会う。それが世界だ。

旅に出ると，いいことと悪いことがある。それが人生だ。

Have a good trip!

向の拡大率を－1.0にして，Z軸回りの回転がそれぞれ，－30度，－60度となります。

最後にカーブの向こうに「STRAI.SUF」を1つずつつなぎます。平面図の範囲が狭いので，テンキーの「＝」をもう一度押してから指定してください。

位置（4000，－4000，0），Z軸回転90度
位置（－4000，2000，0），Z軸回転90度

となります。

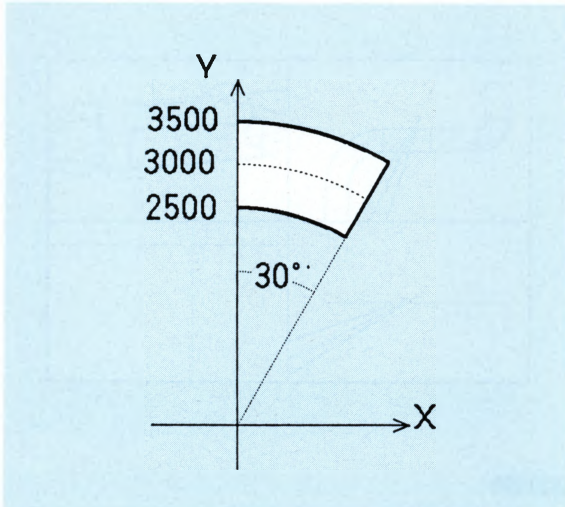
以上でコースが完成しました。とりあえず，この段階で一度セーブしておきましょう。「ファイル/SAVE/フレームソース」で「ROAD1」とでもしておいてください。

○モーションデザイン1

モーションデザインのほうは，特に注意するようなことはありませんので，簡単に解説しましょう。

使用する車種はウイリアムズ，ベネトン，ジョーダン，ティレルの中からお好みのものを選んでください。本文ではウイリアムズで解説します。どの車種を選んだとしても，モーションデザインの段階ではシンプル版の形状データ（頭文字が「S」で始まる）を使用することだけは覚えておいてください。前半のカット「X02A.FSC」は，以下のように設定します。

[1フレーム目]

・「光源設定」でカラーはそのまま，ベクトルをX＝－2，Y＝－3，Z＝－4ぐらいに。「決定」で確定

・「物体設定/追加/SWILL.SUF」で入力モードに入り，

SWILL.SUFの位置：(－4200，2250，0)
Z軸回転　　　　：－70度

として，「決定」「終了」で物体設定モードから出る

・「視点設定」に入り，

視点の位置　：(－500，1000，200)
注目点の位置：(－4100，1700，0)
画角　　　　：15度

として，「作画」で図11のようになっていることを確認し，「決定」する

[15フレーム目]

・「フレームNo.設定」で，15を入力して「決定」

・「物体設定/変更」で画面上の「SWILL.SUF」を指定，「決定」で入力モードに入り，

SWILL.SUFの位置：(－3400，500，0)
Z軸回転　　　　：－30度

図8　「R30.SUF」の座標

図9　位置は同じ，回転角度のみ異なる

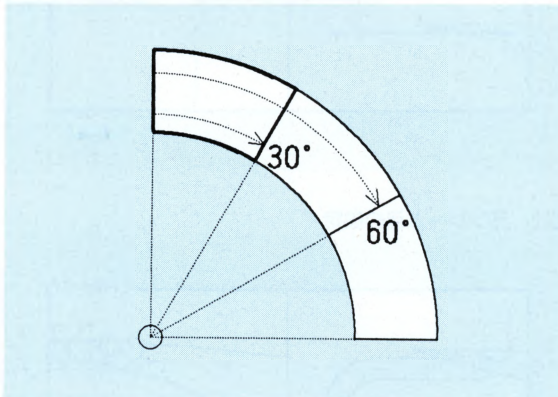
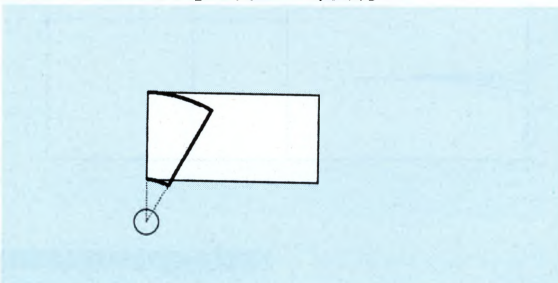



図10　「R10.SUF」の間違った置き方

CGAマガジンの申し込み方

CGAマガジン創刊号はソフトベンダー「TAKERU（武尊）」で入手できます。ネットでもダウンロードできるようにしたいのですが，容量が大きすぎるので検討中です。

地元にTAKERUを置いている店がないという方は困ってしまうかもしれません。そこで，いちおう当チームからも配布します。……が，この時期はCGAコンテスト，およびそのビデオの発送と，連続して忙しいので，必ず下記の注意を守り，負担をかけないようくれぐれもお願いいたします。

・直接申し込むのは，あくまでタケルでの入手が困難な方のみ
・申し込み方法は「現金書留」のみ
・必ず，自分の住所，氏名を書いた宛名シール（フロッピーのラベルなどでもいい）を同封すること
・手間を省くため，宛名シールの氏名は「行」ではなく，最初から「様」にしておくこと
・申し込み期限は3月末日
・発送予定日は不明（CGAコンテストが終わるまで無理？）
・申し込み先

〒533 大阪市東淀川区5-17-2 102号
DōGA内「CGAマガジン申し込み係」

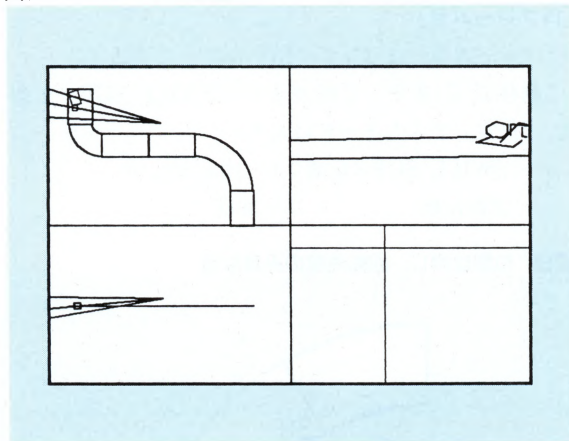
実費は本当は500円程度と予想されますが，そうするとタケルで入手できる人も申し込んできそうなので，1,000円以上の現金を同封していただきます。編集部員に対する手間賃とでも思ってください。

として，「決定」「終了」で物体設定モードから出る
・「視点設定」に入り，

視点の位置　：(−500,1000,200) のまま
注目点の位置：(−3200,380,0)
画角　　　　：15度のまま

として，「作画」で図12のようになっていることを確認し，「決定」する

［25フレーム目］

・「フレームNo.設定」で，25を入力して「決定」
・「物体設定/変更」で画面上の「SWILL.SUF」を指定，「決定」で入力モードに入り，

SWILL.SUFの位置：(0,0,0)
Z軸回転　　　　：0度

として，「決定」「終了」で物体設定モードから出る
・「視点設定」に入り，

視点の位置　：(−500,1000,200) のまま
注目点の位置：(0,0,0)
画角　　　　：15度のまま

として，「作画」で図13のようになっていることを確認し，「決定」する

モーションデザインが終われば,「ファイル/SAVE/フレームソース」で「X02A」と入力します。あとは「終了」でFFEを終わります。

○モーションデザイン2

同様に後半のカット「X02B.FSC」は，以下のように設定します。操作はほとんど同じですので，もう座標値だけで十分でしょう。その前に再びFFEを起動しますが，新しいFFEでは，起動時にフレームソースを指定することができます。

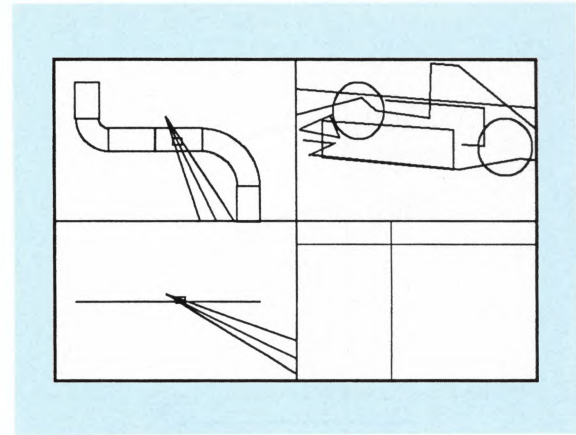
FFE ROAD1 <リターン>

としてください。起動と同時に先ほどセーブしておいた「ROAD1.FSC」がロードされ，コースの設定が終わった状態から作業を再開することができます。

図11　1フレーム目の設定

図12　15フレーム目の設定

図13　25フレーム目の設定

CGAコンテスト事務局より

もうすぐCGAコンテストです。今年は例年に比べ応募作品が多いようで（これを書いているのは締め切り前），内容も期待できそうです。今年も作品集のビデオを作りますが,「ビデオを入手できるのは，CGAコンテストが終わったかなりあとになってしまうので悲しい」とのご指摘も寄せられています。そこで，今年はOh!Xの読者，つまりDōGAの事情にご理解のある方に限り，“プレ申し込み”を受け付けます。

3月中旬までに下記の要領で申し込まれた方には，3月末に発送します。いや，そんなこと書いたらまたトラブルのもとになるので,“3月末に発送できるように努力します”ぐらいにしておきます(まだ，CGAコンテストの日程も決まっていない)。

問題なのは実費が予想できないということです。入選作品が何本で，トータル何分なのかがわからなければ，ダビング料も算出できないし，BGMの著作権料なども見当がつきません。ただ，傾向として昨年より短くなることはまずないので，2,000円では難しいでしょう。ということで，暫定的に2,500円ということにします。もしも，2,000円で収まった場合には，500円はカンパ扱いになりますので，あらかじめご了承ください。

第5回アマチュアCGAコンテスト作品集ビデオ
<プレ申し込み要項>

・申し込み方法は「現金書留」のみ
・必ず，自分の住所，氏名を書いた宛名シール（フロッピーのラベルなどでもいい）を同封すること
・宛名シールは「行」ではなく，最初から「様」にしておくこと
・申し込み期限は3月中旬
・実費2,500円＋任意カンパを同封
・申し込み先

〒533 大阪市東淀川区5-17-2 102号
DōGA内「5th.コンテストビデオ係」

注意：“4th.も送れ”“マニュアルも送れ”などの要求は馬耳東風になることが予想されます。

[1フレーム目]
・光源のベクトルをX＝－2，Y＝－3，Z＝－4
・SWILL.SUFの位置（0,0,0），Z軸回転0度
・視点（－500,1000,200），注目点（0,0,0），画角15度
[10フレーム目]
・SWILL.SUFの位置(2800,－1000,0),Z軸回転－45度
・視点(500,1000,200),注目点(2600,－1000,0),画角15度
[20フレーム目]
・SWILL.SUFの位置(4300,－3900,0),Z軸回転－85度
・視点(800,1000,200),注目点(4400,－3700,0),画角15度

設定ができたら「X02B」としてセーブしてください。

これでモーションデザインは終わりなのですが，せっかくのサーキットなのですから，複数のマシンを走らせたいという希望もあるでしょう。

そういう方のために，もう1台（ここではベネトン）が「X02B.FSC」の後半に画面手前を横切る，というデータも作ってみました。こうしておくと，これに続くカットがウイリアムズでもベネトンでも，自然につながって映像に幅が出るのです。「フレームNo設定」で1フレーム目に戻り，ウイリアムズと同様に，物体追加，変更などで設定してください。

[1フレーム目]
・SBENE.SUFの位置（－1800,200,0），Z軸回転0度
[10フレーム目]
・SBENE.SUFの位置（700,300,0），Z軸回転－15度
[20フレーム目]
・SBENE.SUFの位置(2800,－1000,0),Z軸回転－50度

すでにお気づきだとはとは思いますが，「フレームNo.設定」に入った際に，ナンバーの表示の上下に「▲▼」の表示があります。これはすでに設定されているフレームナンバーを呼び出す機能です。

現在のフレームナンバーが10だとすると，「▲」をクリックするとフレームNo.が1に，「▼」をクリックすると20になります。設定したフレームナンバーをよく覚えて

読者によるほっとけないほっとこらむ

＜Aさん＞9月号の記事のとおり，マニュアルとCGAシステムのディスクを申し込んだのですが，ディスクが入っていませんでした。送ってください。
＜Bさん＞マニュアルが届きました。同じようなディスクが2枚入っていたのですが，どこが違うのですか（ラベルは同じだった）？
うさ子：Aさん，ごめんなさい。そのほかのディスクが足りなかった方も，もう少しお待ちくださいね。それからBさん，ご親切な連絡ありがとうございます。親切ついでに，そのディスクをAさんに送っていただけないでしょうか……，なんて冗談ですよ。
＜あや子＞マニュアルの送金が遅れてしまい，たいへんもうしわけありませんでした。代わりといってはなんですが，東京近辺で人手がほしいときはお声をかけてください。できるかぎりのことはお手伝いさせていただきます。
かまた：もうすぐ，恒例のCGAコンテスト発表会を東京で行います。まだ，正確な日時や場所は決まっておりませんので，詳しくは来月号で紹介します。そこであや子さん，受付を手伝っていただけないでしょうか？　もしよろしければ，当日開場30分前にいらしてください。
うさ子：これであなたも首都圏スタッフ。電子ちゃんと一緒にがんばってください。それではお会いできるのを楽しみにしています。
＜Cさん＞CGAマガジンはコピーフリーと聞いたのですが，DōGAも法人化するというのに，経営は大丈夫なのですか？
うさ子：本当に大丈夫なんですか？
かまた：さあ，どうなんでしょ？　でも，別に無償ってわけではないし，もし運営が成り立たなかったら，有料化するなり，廃刊するなりするだけでしょう。この問題は，この方式に対するユーザーの賛同が得られるかどうかということがポイントですが，私は楽天的に考えています。はっはっは……。ちょっと心配。
＜Dさん＞私のパソコンでF1を走らせたい。ずっと，こればっかり考えているんですが。
うさ子：あなたの夢は，意外と早く実現しましたね。
＜Eさん＞スタッフの方の留年が心配だ。がんばってください。
うさ子：ご心配ありがとうございます。確かにそろそろ試験シーズンです。なのに，ここではディスプレイにかじりついている人が……。
かまた：かなり単位数が危ない者もいますね。私も学生のときは，ノートを借りまくったり，たいへんでした。
うさ子：私はノートの貸し出しに追われてましたけど。
かまた：同じ部員でもずいぶん違うなあ。
＜Fさん＞多忙の折にもうしわけないが，どうかこの私の無理を聞いてくれ！　CGAコンテスト用の作品を完成させるために，PC-9801のRENDを送ってくれ！　同封の為替は輸送費とディスク代だ。余った金は，悪と戦う秘密兵器の開発資金にしてくれ。（「PC-9801のRENDを送ってくれ」係宛の封書）
うさ子：みなさんから寄せられたカンパで，大阪の地下に秘密基地が建造されつつあります。でも，誌面上で公言したら，“秘密”基地にはならないかな。
かまた：それと為替は換金するのが面倒でいやだなあ。なんでもええけど，最近中身を見なくても用件のわかる宛名が流行っているような気がする。
＜Gさん＞やはりFFE.Xを強力にしてください。現在のものも工夫しだいでかなり使えますが，パーソナルCGAシステムとして最も重要なソフトだと思うからです。
かまた：新しいFFEはだいぶ使いやすくなったと思います。モーションデザインはいちばん大切なのに，その手のソフトがほとんどないのは私も不思議です。
うさ子：プログラマの皆さんはいろんな手法のソフトに挑戦してみてください。
＜Hさん＞プロジェクトチームDōGA様へ ら時だつ時5面 付録ディスクが1.4M←いがいにもこんな物だけでいやになっんが多いのでわWINDOSやVSでわうごきません。ZZ うちらにわとうていえんがない。1.4Mのつかい作 その1 DōGA CGAシステムをさしあげる {0} KEYでBATからのがれる。
……以下2ページ続く
うさ子：ひととおり読んでみましたが，意味がよくわかりません。私の頭も混乱してしまいました。
かまた：どれどれ。うっ，たっ確かに。文の途中でいきなり「。」で終わっているどころか，漢字を書いている途中で文が終わっているのはちょっと。そんなことされると，いったいどんな字を書こうと
うさ子：かまたさん，へんなマネしないでください。
＜Iさん＞これは大阪大学コンピュータクラブに対してですが，情教のNeXTのNewsbaseにcomp.X68000.DōGAなるグループを作ってもらえないでしょうか。
うさ子：どなたかと思ったら，うちの大学のI回生ではないですか。
かまた：アンケートに“大阪近辺に在住なので，雑用要員に登録する”と書くひまがあったら，素直に部会へ参加してください。ただ，この時期はクラブはシーズンオフ(?)なので，プロジェクトルームに来るか，4月の新歓のときでも結構です。
＜Jさん＞私はマニュアルを1冊のバインダーに綴じています。筋肉トレーニング，マクラ，押し花といろいろ便利です。
うさ子：そのほかにも，漬物石にしたり，たき火をして暖をとったり，えっと，えっと。
かまた：あの～，CGAの勉強にも使えるのですが……。
＜Kさん＞友人にCGAについて教えようとしたが，すべてしゃべったあとに「で，それって何？　わかんない」といわれてしまった。悲しい。ウルル。
かまた：それは，教え方が悪いのです。正しくは，マニュアルを片手に街頭に立って，道行く人に“アナタハ，CGAヲ信ジマスカ？”と声をかけ……。とにかく，がんばれ。

いないときに便利です。

○作画・アニメーション

モーションをWIREVIEWで確認する場合は，

FF X02A
FF X02B

を実行したあとで，

WIREVIEW /V STRAI.SUF R30.SUF R10.SUF SWILL.SUF X02A.FRM
WIREVIEW /V STRAI.SUF R30.SUF R10.SUF SWILL.SUF SBENE.SUF X02B.FRM

とします。

作画は，

REND /A2 /G STRAI.SUF R30.SUF R10.SUF WILLI.SUF ROAD.ATR WILLI.ATR X02A.FRM
REND /A2 /G STRAI.SUF R30.SUF R10.SUF WILLI.SUF ROAD.ATR WILLI.ATR BENET.SUF BENET.ATR X02B.FRM

とします。「/A2」はアンチエイリアスによって画質をよくするオプション，「/G」はスムーズシェーディングをかけるオプションで，ともに画質は向上しますが計算時間は非常に長くなります。急ぐ場合は省略してください。また，ここではウイリアムズ，ベネトンといった形状データはシンプル版ではなく，本当のデータを使う点にご注意ください。

アニメーションする前には，

CRD X02A /OX02A
CRD X02B /OX02B

で，各画像データを256色に落とします。

また，

MKTCH X02A001 X02B002

で2つのアニメーションをつないだタイムチャートファイル「X02A.TCH」を作成します。ここで，

MKTCH X02A001 X02B001

としないのは，「X02A025.PIC」と「X02B001.PIC」は同じ画像データだからです。

ここまでやれば，

HANIM X02A

でアニメーションします。どうですか？ ……背景がないとどうもイマイチですね。そのへんについては次回で詳しくやるとして，今月修得した知識でコースを用意し，F1を走らせ，いろんなカットを作ってください。

かっこいいカットができたらモーションデータだけでも結構ですから，CGAマガジン編集部に送ってください。「F1追加特集：モーションデータ集」というのをやるかもしれませんよ。

おわりに

さて，この原稿をここまで書き上げた時点で，CGAマガジンはやっぱりまだ影も形もできていません。先月号で予告したとおり，12月20日にTAKERUで姿をお見せするのはきわめて難しいと思います。

その場合は“わざわざ買いにいったのに入ってなかった！”という方も多々いらっしゃると思います。まことにもうしわけありません（あらかじめ，あやまっておくヤツ）。

でも，いくらなんでも，この号が発売されるころには出ているはずです。いまから制作に入れば，春休みに仕上げをして，4月にはデモの1つや2つ完成しているはずです。CGA関係サークルの方々は新入生勧誘対策にでもご利用ください。

柚姫のあちゃちゃちゃCGA

お久しぶりの柚姫です。長いことお休みして，本当にどうもごめんなさい。このコーナーはその名のとおり，CGAのコーナーのはずだったんだけど，ずっとお休みしていたので何を書いたらいいんだか。

姫はこの頃ますます忙しくなって，毎日ばたばたと走り回っています（おかげで少しやせました）。このあいだは九官鳥の実験（脳に電極を……，ちょっと書けない）でスプラッタを見ました。さすがに直後はご飯が喉を通らない。お医者さんなんてこんなのを毎日見てるんだものなあ，すごいというか，なんというか。

それから昨日は，遺跡の発掘現場にお骨を拾いにいったり，となかなか変化に富んだ毎日でなかなかGOOD(!?)。今回は，近況報告ということで，そんな話を少々。

何年か前から，伊丹空港の周辺の再開発が進められてきているんだけど，ここには有岡城址（織田信長に滅ぼされた）など数多くの貴重な遺跡があって，それらを再開発の前に調査しています。姫はその近くのお寺から出てきた古人骨を見にいってきました。古人骨といっても17世紀以降のものなので，あんまり古くはないので少し残念だったけど，ちょっとドキドキ。

発掘現場はいろんな出土品が所狭しと並べられていて，なかなか活気に溢れていました。人骨もたくさん出てきていて，その中から2つ，3つ（2，3体分）を見せてもらいました。火葬にして埋めてあったものなので，かなりバラバラ。たくさん穴が開いてスカスカで，触れ合うとシャランといい音がしました（貝でできた風鈴のよう？）。

一緒に行った京大の先生はそのバラバラになった骨を見て，“これは大腿骨，これは頭蓋骨の耳の奥の部分”と当てていくのですが，なんかジグソーパズルみたいで面白かった。自分の予想が当たったり，パズルが組み上がっていったりすると，もううれしくてうれしくて（人の骨で遊ぶなって？）。そのうえ先生は，性別やおおよその年齢，死因などを当てたりして，姫はすっかり感心してしまいました。自然人類学ってロマンだなあ，なんてね。

発掘現場の人は女の子がうれしそうに骨を触っているのを見て，ちょっと驚いていたようですが，そんなにめずらしいかな？ でも，お葬式のときに焼け具合を指定したり，つい骨を組み立ててしまいそうで怖いなあ。

早いもので，もう新年です。姫にもいろいろと今年の抱負があるんですが，やっぱりなんといっても，今年は「1年間を元気に楽しく過ごす」というのが第1の目標。去年は初夏には事故にあって半月ほど松葉杖をついていたし，夏の終わりや秋にも体をこわしたりして，なかなかたいへんでした。

CGAのほうも，当初の目標からはまだまだほど遠くて，やっとCADで箱を作ったところ。春までにはうさぎさんを作って，これらでなにか小さな作品を作ってみたいんだけどなあ。まだあんまり使えていないという人も，姫と一緒にがんばろうね。

各種ツールを使ったモデリング(2)

文月 涼

■前回のおさらい

12月号ではまた来月などといいながら，DōGA連載のコラムゆえ，1月号は本編と一緒に休んでしまいました。

さて，前回のクラインの壺を実際に作ってみた方はさぞや悩んだことと思います。記事の欄外，および写真の補足説明では不十分であったと反省しています。

TUBEはSUFファイル上に存在する面の順番に筒を作っていくのではなく，最初の面とその面に最も近い面を探し，その面間で筒を作り，以後同じ処理を繰り返していくのです。したがって，クラインの壺の断面図のように，複雑に面が交差している場合，TUBEは人間が考えているような結果を出してはくれません。

このようなケースで断面を10個使用しているとした場合，たとえば1から5までの断面と5から10まで（あの壺ではスタート＝ゴールなので，1と10は同一面）の断面を含んだ2つのSUFファイルを用意し，別々にTUBE処理をかけ，その2つの結果をCADに連続して読み込んで合成するか，FFE＋KAMAで合成し，目的の物体を生成するのです。実際に，クラインの壺では4つ程度に断面図を分割して，最終物体を生成しました。

■車を作る，の続き

前回，車を作る話の途中で終わってしまったのですが，その過程で，一部から私の物体のモデリングの手法はまだ読者の間で一般的でないという指摘がありましたので，ここで私がスタンダードとしているモデリングの手法を挙げてみましょう。

1)　CADで打つ

便利なツールのなかった時代は，物体の図面を書き，すべてをCADでしこしこと打ち込んでいました。いまでもときどきはやりますが，思い出してもぞっとしてしまうあの日々といった感じです。

2)　断面図→TUBE

モデリングしたい対象が断面図を打ち込める筒状である場合，最も効率のよい方法です。というより，現在はこの手法が使えないか，まず最初に考えます。しかし，モデリングする物体が，必ずしも断面図を平面で表せるとはかぎりません。そういう意味ではたいへん汎用性の低いモデリングの手法です。しかし，前回チョロっと書いたように，裏技「TUBEで断面として使う面は必ずしも全点が同一平面上にある必要はない」を使うことで，汎用性はぐっと高くなります。

3)　KAMA.Xを使う

DōGA内の俗語で「KAMAる」といわれる作業です。タイヤとボディを一体化させたいときなどに，「KAMAって」という表現で使用します。KAMAは複数のオブジェクトをFFEで指定されたとおりの形でひとつの物体として合成するツールです。これもイメージを広げる素材であると考えます。

＊　＊　＊

読者の方々はだいたいの場合，モデリングというとまずCADに向かい，数時間のうちにげんなりしてしまうことが多いようです。そういったときは，対象物を全部手で打とうとするのではなく，断面をCADで作り，TUBEで外側を作るといったショートカットを使い，なるべく無駄な労力を消費しないように考えるといいでしょう。

また，以後私が「TUBEして」「KAMAって」といった場合は，上記のような意味なので，そう理解しておいてください。

■重要な線

さて，いよいよ前回の続きなのですが，TUBEで断面図を取りやすい単位に車を分割して考え，作業はその構成単位で進めることになります。このときに忘れてならないのは，分割した線上の点だけはきっちり1つひとつ座標を決定しておくことです。もともと別個に存在するものを別々にモデリングするのではなく，本来ひとつの物体として存在する車をいくつかのパーツとしてモデリングした場合，要となるのは車をパーツに分割した線です。

この線を（実際には線を構成する各点を）きちんと決定せずにモデリングした場合，各パーツが仕上がったときに組み合わせようとしてもうまく継ぎ目が合わなかったり，バンパーがボディにめり込んでいたりという事態になりかねません。したがって，モデリングに先立っては，その線を図面上などではっきり決定していてください。

さて，実際のパーツのモデリングは，前回紹介したTUBEを使用するコツをふまえたうえで，継ぎ目の線をきちんとおさえた断面をCADで打ち込むところから始まります。断面図はできるだけ詳細に作るのが望ましいのですが，感覚的に難しい場合はあと回しにしても問題ありません。

たとえばバンパーの断面図を作っているときに，ボディ内側に入り込んでいるダクトを作ることはほとんど不可能です（面を構成しない点の集合を使えばできるときもある）。作ろうとするバンパーを垂直に切って断面図を作っているのに，その断面に平行なダクトはいたずらに断面を増やすことになるだけだからです。こういった場合はダクトが存在する部分はただの平面にしておいて，あとからCADでダクト部分だけを作るのです。

それからもう1点。左右対称である物体を作っているときは，左端あるいは右端から中央までの断面だけで十分，ということです。中心から片側だけの断面にTUBEをかけ，そのうえでMIRRで反対側を生成させるのです。

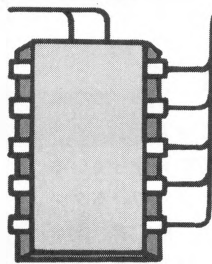
さまざまな工夫により，労せずしてバンパーの外側を作ることができました。次にバンパーの細かい部分を作り始めます。たとえばダクトなどのエアスクープですが，最初にTUBEで生成された外側の所定の部分に穴を開けます。

TUBEは断面図に忠実に筒を作りますので，バンパーの床部分や，ダクトが開くべきところなど，本来は必要がない部分も作ってしまいます。この部分をCADで指定して順に面削除していくのです。外側から必要な部分のみを切り出し終わったら，バンパーからへこんでいるエアスクープダクトや，ちょっとへこんだウィンカーなどを順に，手作業で作っていきます。

バンパーにエンブレムなどがつく場合は，エンブレムだけを先に作っておいて，KAMAで合成するという手もあるのですが，装飾品はあとでまとめてやったほうがいいでしょう。このとき，ウィンカーのカバーレンズや色が違う面は，面のアトリビュートも変更しておきます。こうして，分割した各パーツを仕上げていきます。

バンパーの装飾品をCADで打つのもいやだという人は，分割した各パーツをさらに細かく割って作ってもいいのですが，たかが四角いダクトであれば，前回にも書いたとおり，CADの得意とする機能のひとつなので，CADでやってみるといいでしょう。

来月はパーツのチェックと合成です。

ハードウェア工作入門［32］

コンピュータアーキテクチャ編

減算器の設計

Misawa Kazuhiko
三沢 和彦

今月は減算回路の設計を行います。加算をうまく操作することによって，減算を実現するまでの手順をこと細かに説明していきます。以前学習した論理演算も出てきますので，忘れている人はしっかり思い出してください。

前回までで加算器は，完全にマスターしたといってもよいでしょう。1桁の加算器は，XOR回路そのものであることがわかりました。

そして実用的な加算器を設計するには，1桁加算器をベースに繰り上がりのAND回路を組み合わせていくだけでOKです。加算器を実現するための論理回路についても細かく設計してきましたが，最終的には実際の回路としてTTL ICのシリーズに加算器のパッケージが用意されているので，それをそのまま使えばいいようになってます。

また，2つ以上の数の加算を行うには，演算結果を格納しておくレジスタを用意しておき，次の数を足すにはそのレジスタの中身に順次加えていくような形にしていけばよいことも理解できたと思います。レジスタはフリップフロップという回路からできていて，外部からクロック信号を与えることによってデータをセットすることができます。

さて，今回からは数値データを処理する演算回路を少しずつ発展させていくことを考えます。数値データの演算には，加減乗除の四則演算があります。このうち最も基本なのは加算であり，これはもうマスターしたといえるでしょう。そして，ほかの演算は加算を変形していけば，実行できるのです。ここでは，減算を考えてみましょう。減算においては「引く数を負の数に置き換えて加えてやる」と考えると，これも加算の一種になるのです。たとえば，

8－3＝5

は，8から3を引く減算ですが，引く数の3を負の数－3に置き換えて，

8＋(－3)＝5

と考え直してやれば，8に－3を加える加算と見なすこともできるのです。ですから，2進数でも負の数を表現できるようにしておけば，これまでに設計製作してきた加算器を少し発展させて減算もさせることができます。

そこで，今月からは，減算器の設計製作に移ってみたいと思います。まずは引く数を負の数に変換する回路について考え，次にその変換回路を加算器と組み合わせる方法と，最後にはひとつの回路で加算と減算とを必要に応じて切り替えて実行する回路を設計製作していく予定です。

2進数における負の数

では，最初に2進数で負の数を表現する方法を考えましょう。前回の加算器では2桁の計算でしたが，今後は4桁の計算を扱っていくことにします。さて，負の数を考えるうえでキーポイントになるのは，

1111＋0001＝10000……(1)

という計算です。この2進数の計算を10進数に直すと，

15＋1＝16

ということになります。

しかし，ここで，ちょっと見方を変えて，答えのいちばん上の繰り上がりを無視してみましょう。

1111＋0001＝0000

これを10進数に置き換えてみると，1111＝－1と解釈すれば，

(－1)＋1＝0……(1')

という計算を行っているのと同じことになります。

同様にして，足し合わせると答えが10000になる組み合わせを並べてみましょう。

1110＋0010＝10000……(2)
1101＋0011＝10000……(3)
1100＋0100＝10000……(4)
1011＋0101＝10000……(5)
1010＋0110＝10000……(6)
1001＋0111＝10000……(7)
1000＋1000＝10000……(8)

10000→0と考えて，10進数に置き換えていくと，

(－2)＋2＝0……(2')
(－3)＋3＝0……(3')
(－4)＋4＝0……(4')
(－5)＋5＝0……(5')
(－6)＋6＝0……(6')
(－7)＋7＝0……(7')
(－8)＋8＝0……(8')

以上の結果より，

1110＝－2……(2")
1101＝－3……(3")
1100＝－4……(4")
1011＝－5……(5")
1010＝－6……(6")
1001＝－7……(7")

と対応させることができます。ここで，

1000＝－8……(8")

とするのは少々問題があります。というのも，(8)式を見てわかるとおり，

1000＝8

でもあり，1000は－8と8のどちらなのか区別がつかなくなってしまうからです。

この問題に関連して，最初のところで出てきたように，1111＝15＝－1となり，一体どちらなのかわからないという問題もあります。そこで，もう一度，(1)～(7)式と(1")～(7")式とをにらめっこすると次のような規則性に気づくと思います。つまり，

「10進数の負の数に対応しているのは，4桁目が1の数である」

という規則です。そこで，4桁目の最上位ビットを符号ビットとして，そこが0なら正，1なら負と約束することにします。この約束に従えば，1000＝－8，1111＝－1，と一義的に決まります。

以上で，加算と減算とを組み合わせる約束が決まりましたが，正の数から負の数へ変換するにはどうしたらよいでしょうか。

0001＝1←→1111＝－1……(1"')
0010＝2←→1110＝－2……(2"')
0011＝3←→1101＝－3……(3"')
0100＝4←→1100＝－4……(4"')

0101＝5 ←→ 1011＝－5……(5''')
0110＝6 ←→ 1010＝－6……(6''')
0111＝7 ←→ 1001＝－7……(7''')

ここでいう，「正の数から負の数へ変換する」というのは，たとえば5に対応する数値データの0101を処理して－5である1011を得る論理回路をどう設計するかということをいいます。この点について次に考えてみたいと思います。

負の数と補数表現

正の値を負の値に変換していくには，次の手順を踏めばわりと簡単に理解できます。まず，変換前の数値データの各ビットについて0と1とを反転させます。

そして，反転後のデータにそれぞれ1を加えるのです。

	(反転)	(＋1)
0001 →	1110 →	1111……(1'''')
0010 →	1101 →	1110……(2'''')
0011 →	1100 →	1101……(3'''')
0100 →	1011 →	1100……(4'''')
0101 →	1010 →	1011……(5'''')
0110 →	1001 →	1010……(6'''')
0111 →	1000 →	1001……(7'''')

すると，あら不思議，(1''')～(7''')の結果と同じになっているのです。

以上のように取り決めた負の数を「(2の) 補数」と呼んで，コンピュータの演算では非常に一般的なデータ形式になっています。この2の補数表現を使えば，減算と加算はまったく同じ回路を使うことができるので，非常に便利になっています。

では，補数へ変換する手順を論理回路で実現するとしたら，どうしたらよいでしょうか。それには，基本的に各ビットの0←→1を反転させる論理演算を考えなければなりません。先月までに出てきた基本的な論理演算には，AND，OR，NOTの最も基本的な3種類と，次に基本的なXORとがありました。これらの論理表を表1に載せますが，それを見るまでもなく，NOT回路が0←→1の反転そのものである，ということがわかると思います。

そこで，図1のような回路のブロックを考えてみました。これは基本的に前回までの加算器を流用し，加算器の片方の入力にはその前に反転用のNOT回路をはさんでおいたものです。

また，2の補数を作るためには，最後に1を加えなければならないのですが，それには，最下位（1の位）の加算器も繰り上がり（キャリ）つきにして，あたかも下の位（1の位の下になるから，実際にはない）から繰り上がってきたかのようにして1を足し込んでやればOKです。また，加算器のときのようにレジスタを使って，複数回の演算を順次行うようにすることもできます。いま扱っている減算は加算そのものなので，レジスタの使い方は加算器のときとまったく同じでかまいません。

加減算の切り替え

これで減算器の基本的な部分はできましたが，これだけではただ減算しかできません。実際のCPUでは加算と減算との両方ができるようになっています。

まず考えられるのは，図2のように加算器と減算器とを別々に用意することですが，減算器が加算器をほとんどそのまま流用していることから，図中の囲んだ部分はまったく同じ回路が2つ並んでいることになります。

さらに問題なのは，加算器と減算器とが別々になっているために，演算する2つの数値データをCPUに入力するときにデータを別々の入力端子に入れなければならず，また演算結果も別々の出力端子から出てくるという点です。同じバスラインデータを流すためには，加算と減算で入出力を切り替えなければなりません。

そこで，加算器の回路部分と入出力端子を共通にした形で，必要に応じて加減算が切り替えられる回路を設計したいと思います。図2のブロック図で，共通でない部分に着目してみましょう。すると，減算のときに2の補数を作るために，

1) 入力データの各ビットが反転する部分
2) 最下位ビットに1を足し込む部分

の2カ所であることがわかります。この2カ所の回路を連動して切り替えられるようにしておかなければなりません。このときの切り替え信号の与え方は，加算/減算を論理レベルのH/Lに振り分けることにします。

では，この切り替えを実現する回路を実際に設計してみましょう。まず，反転部分のブロック図を図3に示します。入力，出力ともに1本ずつで，このほかに制御信号を1本設けます。ここで制御信号がL（0）

図1　減算器のブロック図

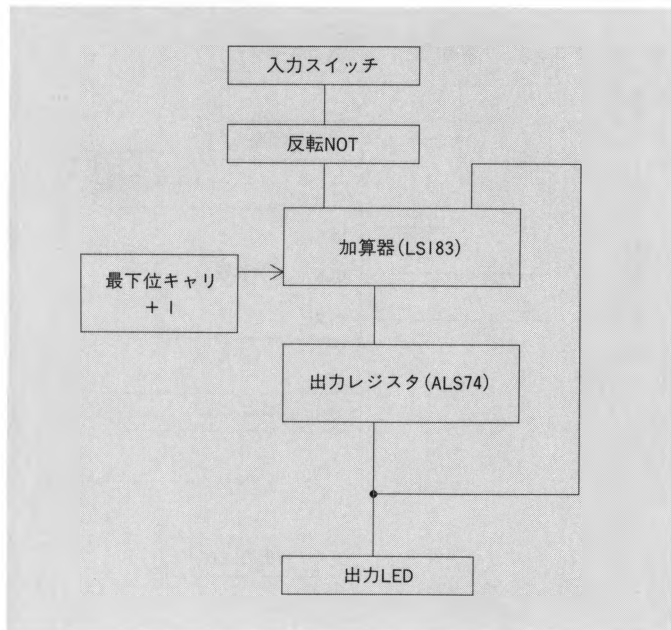



図2　加減算器の組み合わせブロック図

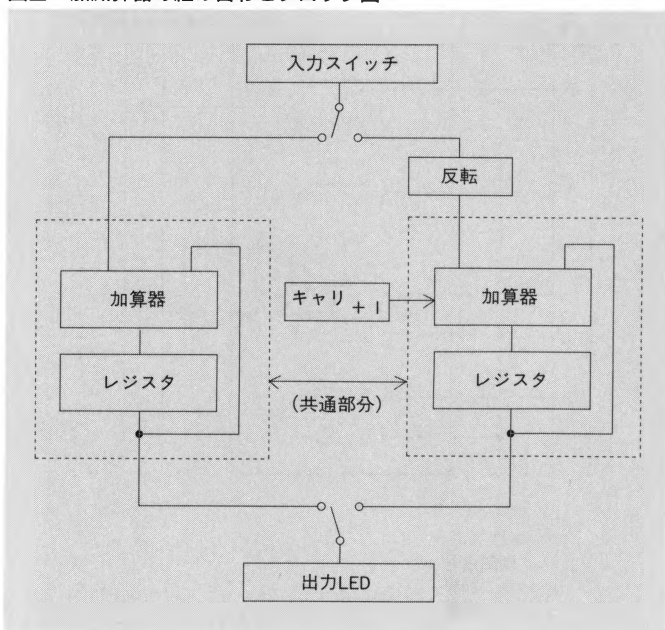
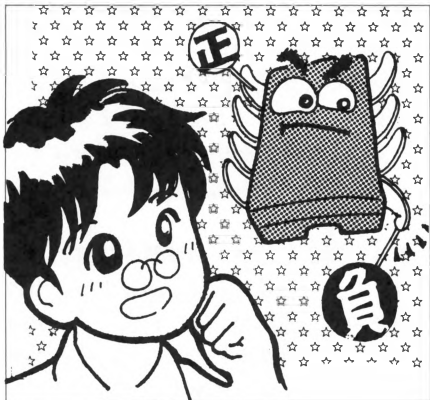



illustration:Y.Kawahara

のときに加算を実行し，H（1）のときに減算に切り替えるように約束することにします。

それには，1ビットのデータに対して，制御信号が1のときに反転し，0のときはそのまま素通しにするようにします。この論理演算を，縦に制御信号，横に入力データをとって論理表を書いてみると表2のようになります。この論理表を見て気がつきませんか？　この論理演算は，加算器の設計のときにも出てきたXORの論理演算とまったく同じになっています（表1）。もしデータが4ビットであれば，4個のXORを並べて，制御信号を共通にすべてのXORゲートの片方の入力に入れてやればよいことになります。この回路を図4に示します。

次に最下位ビットへの1の足し込みですが，加算のときには0，減算のときには1を最下位への繰り上がり（キャリ）入力に入力してやればよいことになります。制御信号がL（0）のときに加算，H（1）のときに減算という約束なので，ちょうど制御信号をそのまま最下位ビットへの足し込みデータとして入力することができます。なお，加算器回路には，前回と同様に既成のパッケージを使うことにしますので，回路図は簡単に図5のようなものになります。

以上の回路をひとまとめにすると，図6のようになります。

切り替え式加減算器の実際

では，図6のブロック図をもとに，実際のTTL ICを使った回路を設計していきましょう。先ほど述べたように今回は4桁の2進数データを扱いますので，これまで扱ってきた加算器とレジスタ，及び今回追加した反転回路をすべて4ビット回路に置き換

表1　基本論理演算

AND

	0	1
0	0	0
1	0	1

OR

	0	1
0	0	1
1	1	1

NOT

0	1
1	0

XOR

	0	1
0	0	1
1	1	0

表2　制御信号つき反転回路論理表

制御信号＼データ	0	1
0	0	1
1	1	0

XORと同じ

図3　制御信号つき反転回路

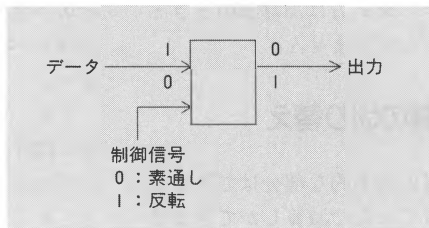



図5　加算器の回路図

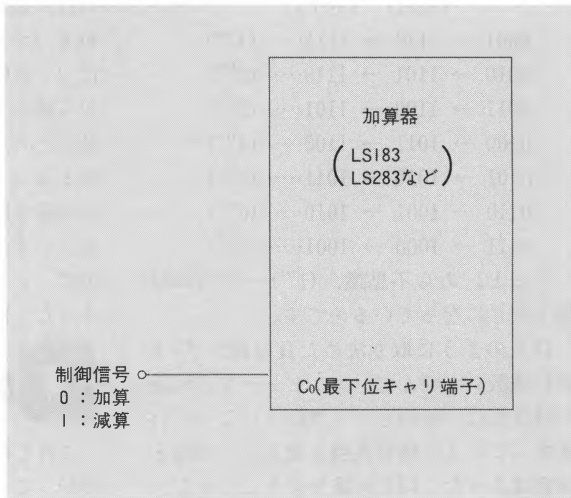



図4　4ビット反転回路

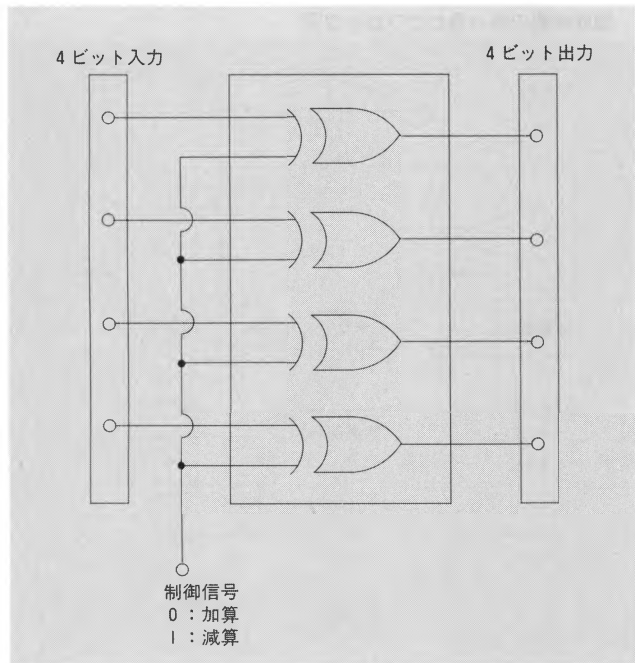
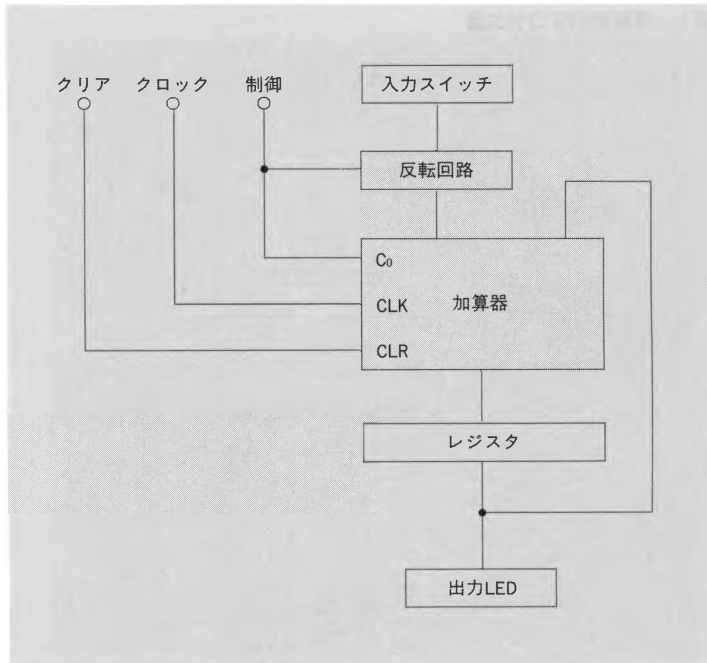



図6　切り替え式加減算器のブロック図

えます。

　加算器　　LS183 → LS283
　レジスタ　ALS74 → ALS175
　反転回路　LS86

　それぞれのICの規格が図7です。ICの特徴を変更前のものと比較しながら，述べていきたいと思います。

　LS283は4ビットの繰り上がりつき加算器です。LS183との違いは，LS183が2ビット回路でしかも繰り上がり入出力端子がすべて独立についているのに対して，LS283は繰り上がりが最下位ビットへの入力と，最上位ビットからの出力としか出ておらず，途中の桁上がりに関しては，パッケージ内に組み込まれてしまっている点です。4ビット入力A1～A4とB1～B4の2系統，4ビット出力Σ1～Σ4，最下位キャリ入力C_0と最上位キャリ出力C4とがLS283の入出力端子になっています。

　ALS175は4ビットDフリップフロップで，基本的にはALS74に入っているDフリップフロップと同じものが4個入っています。ALS74との違いは，ALS74ではクロック端子とクリア端子とが2個独立になっているのに対し，ALS175ではすべて共通になっています。先月のレジスタ加算器の回路を見てわかるとおり，クロックとクリアは各ビット共通で直結になっているので，ALS175を使えばその部分の配線をパッケージの外部で行う必要がなく，手間が省けます。前回の回路でもこのALS175が使えたのですが，まずはより基本的で使用頻度の高いALS74を先に扱ってみました。

　また，このALS175にはALS74にはあったプリセット端子がついていません。プリセットというのはリセットの逆で，出力Qを強制的に1（リセットではQを0）にします。今回の回路ではプリセットは使用しないので，ALS175でもかまいません。

　LS86は独立したXORゲートが4個入っているパッケージです。すべて端子が独立なので，今回の4ビット反転回路では各ビット共通になっている制御信号をそれぞれ外部で配線しなければなりません。

　図6のブロック図を回路図に書き直したものを図8に示します。しかしながら，実はこの回路は完成ではありません。加算器回路のときは，このままで完成としていたのですが，加減算では出力に負の数も現れてきます。ところが，2の補数というのは一見しても10進数に直していくつになるかがわかりづらくなっています。そこで，来月は出力LEDの部分を大幅に手直しして，演算結果が見やすいように工夫したいと思っています。

　それではまた来月まで。

図7-1　LS283規格表

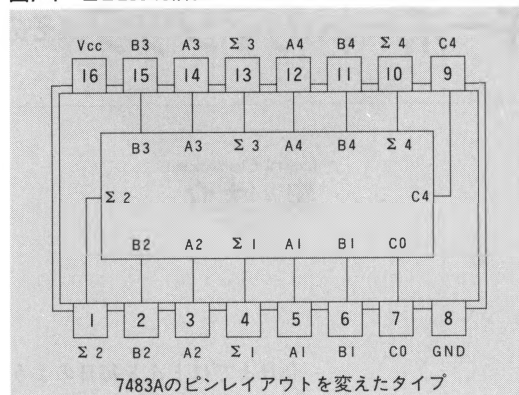
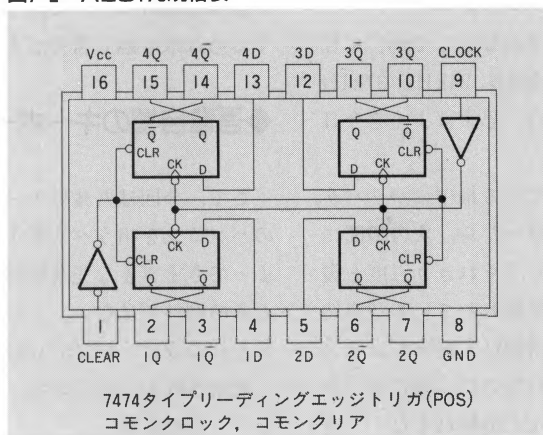
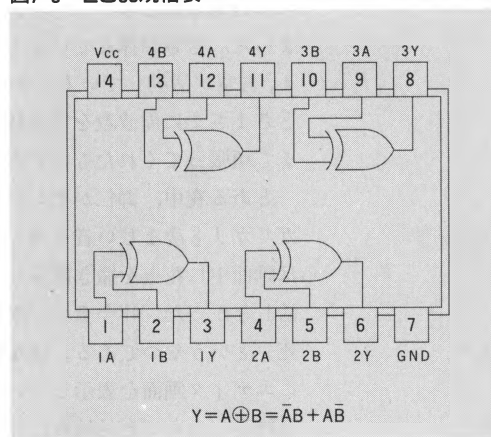
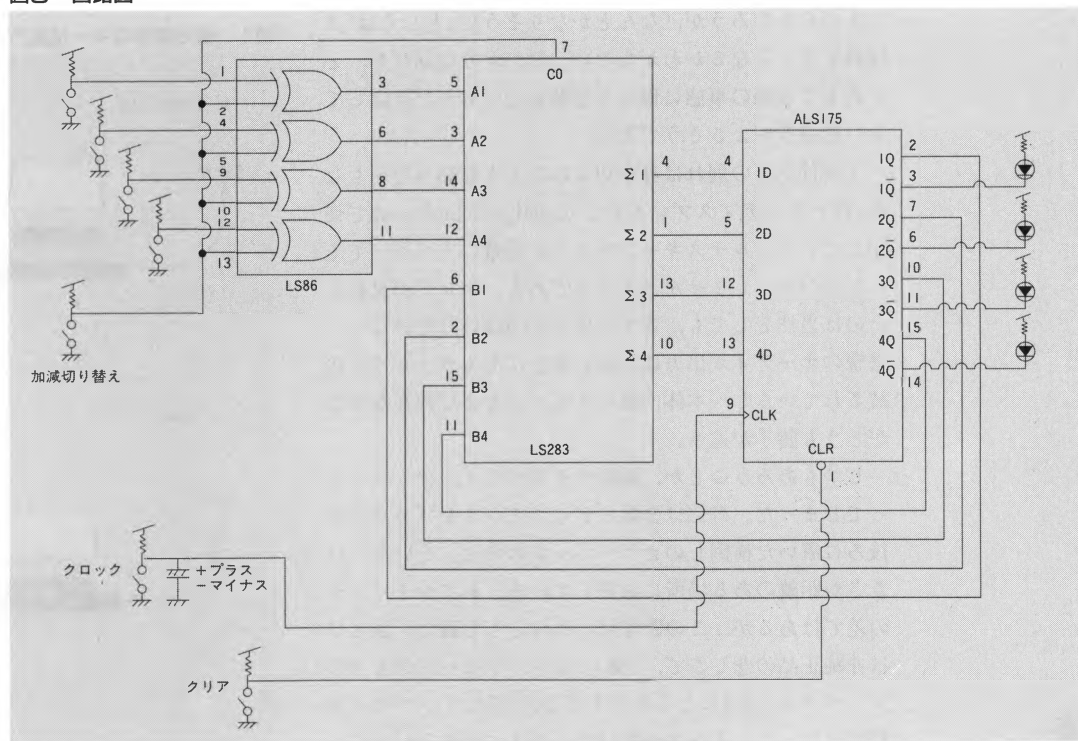



図7-2　ALS175規格表

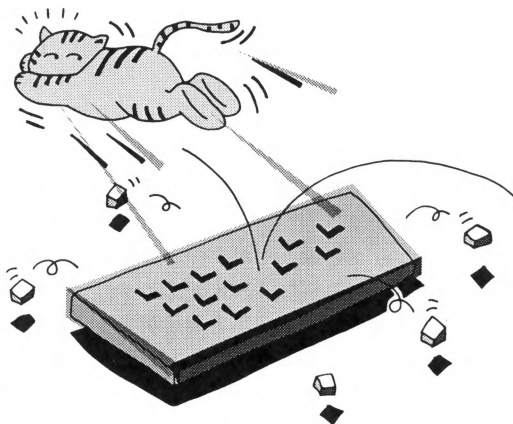
図7-3　LS86規格表

図8　回路図

吾輩はX68000である

［第20回］

キーボードのマジック（その1）

Izumi Daisuke
泉　大介

吾輩と御仁を結ぶ
絆
それがキーボードである

今日までほとんど毎日のようにうちの御仁にお仕えしてきたが，積み重なる歳月の重みに耐えかねたのか，ついに吾輩の顔ともいうべきディスプレイが音を上げてしまった。吾輩が休んでいるときもテレビとして御仁に仕え，吾輩が活動しているときには15kHz, 24kHz, 31kHzとさまざまに周波数を切り替えられ，思えばよくぞ今日まで頑張ってくれたものである。

とある夜中，御仁がテレビを見ている最中にバリバリ，ガリガリと凄まじい音を発したと思ったら，次の瞬間には画面中に花火を撒き散らし，そしてそれきり二度と復活しなかった。昨今テレビや新聞を賑わしている「突然死」というやつである。ほんの1時間ほど前までは元気にエディタ画面を表示していてくれたのに，御仁が作業を終了しテレビを見始めたら，なんの前触れもなく，突然に逝ってしまった。さっそくシャープ大人の元へ戻すことになるだろうが，「なんとかなりそうだ」もしくは「ご臨終です」となるかわからない。気がかりな話だが，とりあえず最悪の事態に備えて吾輩も心づもりだけはしておいたほうがよさそうである。

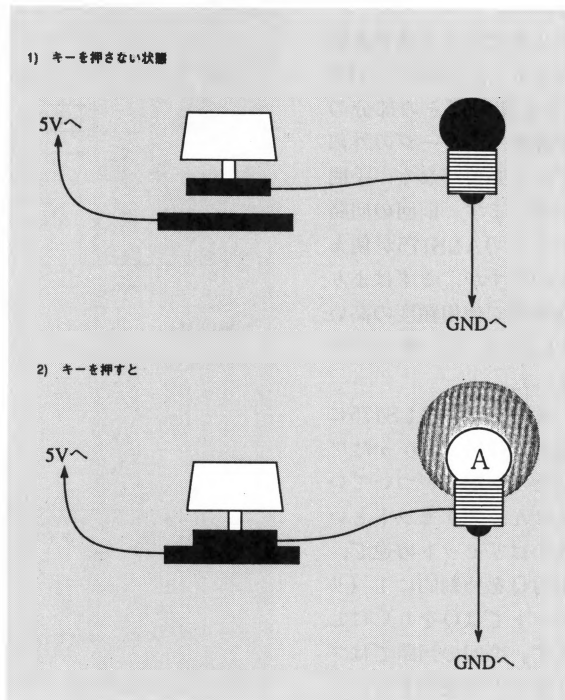
よき伴侶との別れは身を切られるような痛みをともなう。代わりのディスプレイとして，御仁がMacintoshで使用していたマルチスキャンモニタを吾輩に当てがってくれたのだが，しょせんはモニタである。テレビが見れないのは当然としても，音すら出ないのはいただけない。吾輩のオーディオ出力は，同じ並びにあるステレオに接続されているし，本体内蔵のスピーカもまだ健在なのだがどうも勝手が違う。

しかもあろうことか，画面サイズが1インチ小さくなってしまった。御仁は吾輩とディスプレイを「コタツの後ろに置いた横倒しのカラーボックスの上」という，いささか距離のある場所に設置している。わずか1インチの差ではあるが，この距離はいかんともし難い。さらには非純正品の悲しさで，768×512ドットモードのときのアスペクト比を1にしようとすると，実質12インチモニタ程度になってしまうのが視認性の悪さに拍車をかけている。モニタを当てがわれただけでも感謝すべきだとは思っているのだが，ディスプレイテレビの逝去以来，御仁の吾輩を眺める目がなんとなくよそよそしいような気がするのは取り越し苦労であろうか。

◆吾輩自慢のキーボード

さて，今回は吾輩のキーボードを紹介したい。次世代のパソコンを担うべく誕生した吾輩にはシャープ大人によってさまざまな先進機能が装備されたが，キーボードもその例外ではない。その一番の特徴は，スペースバーの左右に設けられた5つのXFキーである。カナ漢字変換で使用されるこれらのキーの効用は，まことに計り知れないものがある。

最近では某国民機用の日本語FEPでもCTRLキーを併

図1　最も簡単なキー接続方法

用して文節の伸縮やカタカナ変換を行えるようになってきているが，吾輩が誕生した当時はファンクションキーまで手を伸ばさなければ，あるいはカーソル移動キーまで手を伸ばさなければ作業できないものが大半を占めていた。左手の親指でチョコチョコと文節を移動し，シフトキーの併用で自由に文節を伸縮できる吾輩の日本語FEPに慣れ切った御仁は，当時某国民機で頻繁に「親指の素振り」を行ったそうである。

キーストロークの深さといい，その重さといい，既存のキーボードの中でも結構いい線いっていると自慢なのだが，ただひとつ吾輩の気になっているのはCAPSキーの位置である。Compactでは大幅な配置替えがあったのだが，従来のテンキーつきキーボードではテンキーの上という最果ての地に配置されているのである。御仁は日本語FEPにFIXERを使用しているので，CTRL＋XF5でCAPSキーをON/OFFしているのだが，そうでないASKユーザーの諸兄はいかに対応なさっているのだろうか。

◆キー入力を感知する

さてそのキーボードだが，どのような仕組を用意すれば何十個とあるキーの中のどれが押されたのかをチェックできるか諸兄はご存知だろうか。キーが数個しかない場合は簡単である。図1のようにスイッチを用意すればいい。ここではメモリの特定のビットが1になるという通常のインタフェイスの代わりに豆電球をつけているが，原理的には同じである。これをキーの数だけ用意すれば，キーボードの完成となる。たとえばこの方法でキーを8個並べてみると図2のようになる。キーボードと豆電球が1対1に対応しているため感覚的にもわかりやすいし，どのキーが押されたのかを独立して判定できるというメリットもある。

しかしながら吾輩のキーボードのようにキー数が100個を超えるキーボードではこのような方法は通常とられない。この調子でキーと豆電球を1対1につないでいくと，膨大な配線が必要になってしまうからである。また，CPUとのデータのやり取りを考えると豆電球をそのままメモリに置き換えることはできず，また適当なパラレルインタフェイスをかませる必要があるという点でもこの方法はいただけない。100ビットを超えるパラレル入力を扱える汎用のインタフェイスなどというものは聞いたことがないし，かといって，そこら中に溢れている8ビットパラレルのものを使用するとなると，10個以上のインタフェイスを使用しなければならないことになる。こいつらがてんで勝手に「キーが押されたよ」とCPUに割り込みをかけ始めた日には目も当てられない。

というわけで，直観的な図2の方法は，本格的なキーボードに使用するには実装上の問題を抱えているわけだ。巷に溢れる8ビットのパラレルインタフェイスをうまく利用するためには，100個以上のキーの状態を8個の豆電球で扱えるようにする必要がある。どなたか，うまい解決策を思いついた方はいらっしゃるだろうか。次に進む前に，ぜひとも一度考えてみていただきたい。

◆キーボードマトリクス

上記の問題を解決するために一般に採用されている方法は，キーボードマトリクスと呼ばれるものである。つまりはキーの行列だ。あたかも数学の行列を思わせる配列にキーを並べるものだが，その原理は単純である。一度に100個のキーの状態を把握しようとするから100ビットパラレルなどという妙ちきりんなものが必要となるの

図2　8個のキーを図1の方式で並べてみる

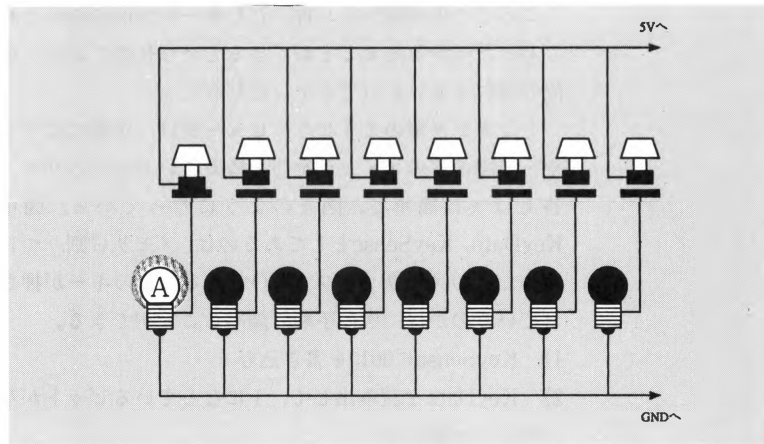



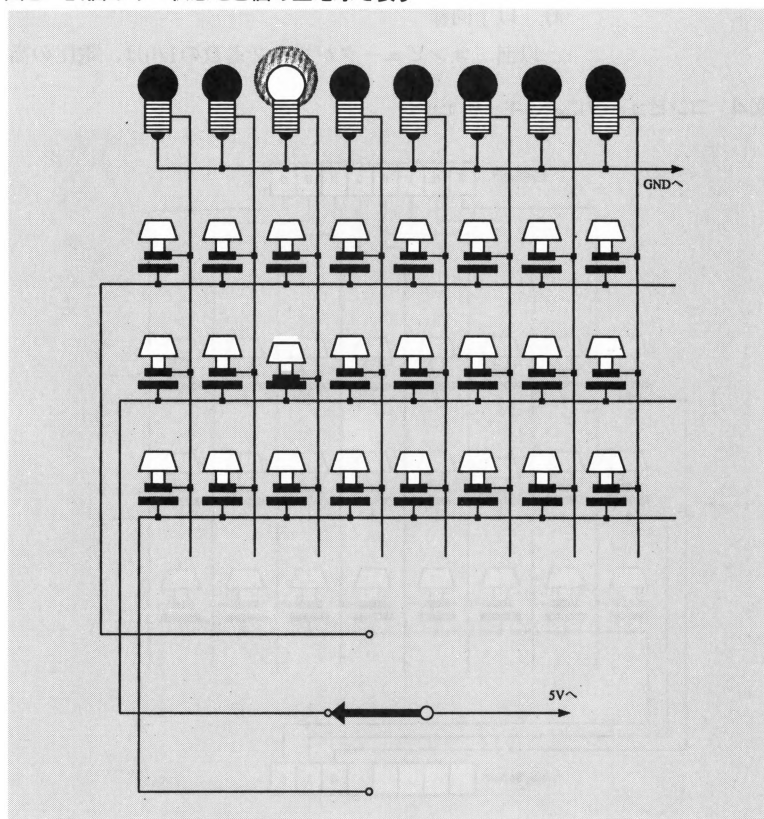
図3　24個のキー状態を8個の豆電球で表す

であって，一度にチェックできるキーの数を８個に限定し，何度かに分けてチェックしていけばいいではないか，というのがその基本原理である。

図３をご覧いただきたい。このようにキーを配置すると，24個のキーがあっても電球の数は８個で済む。画面左下のスイッチは，どの列の８個のキーに電圧をかけるかを選択するものである。押したキーのある列に電圧がかかれば，回路が閉じて対応する豆電球が光るようになっている。もし電圧をかけた列のキーが押されていなければ，豆電球は消灯したままとなる。現在は真ん中の列の左から３番目のキーが押されており，スイッチが真ん中の列に電圧をかけているので豆電球が点灯している。

1)　電圧をかけたのはどの列か

2)　点灯しているのはどの豆電球か

この２つの情報から，押されたキーが24個の中のどれであったのかを知ることができるという仕組である。電流の流れを追いかけてみていただきたい。

ところで吾輩のようなコンピュータは，実際にはどうやって図の下のスイッチを切り替えているのだろうか。答えは実は簡単で，図４のようになっている。図中KeyData，KeySenseとしてあるのは，メモリに割りつけられた例のメモリマップドI/Oである。どのキーが押されているのかは，次の手順で調べることができる。

1)　KeySenseに001_Bを書き込む

2)　KeyDataを読み出して，１になっているビットがないかどうか調べる

3)　KeySenseに010_Bを書き込む

4)　以下同様

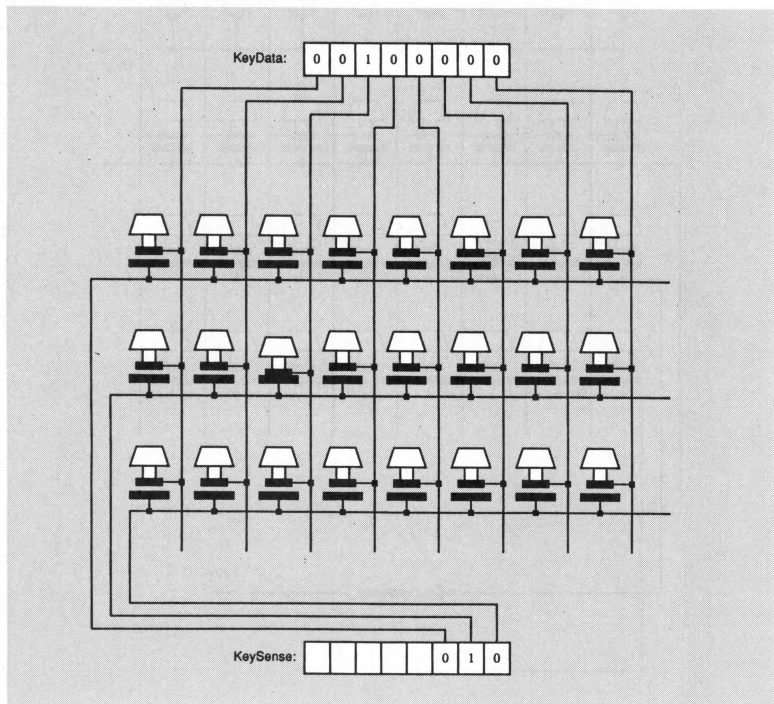
以前，コンピュータが扱う２進数の1/0は，電圧の高/低，電球の点灯/消灯を数字で表したものにすぎないという話をしたことがあるが，ここでもう一度それを思い出していただきたい。1)でKeySenseに001_Bを書き込むと，つながっているラインに電圧がかけられる(１は電圧の「高」を表している)。これは，図の最下列のキーに電圧をかけたのと同様である。もしこのとき，最下列のキーのどれかが押されていれば，KeyDataに対応するビットが１になってレポートされる(１は豆電球の点灯という意味になる)。同様にして，３つの列を順々にスキャンし，押されているキーをチェックするのである。

図4　コンピュータによるキーチェック

◆キーボードとCPU

吾輩がやってくるまで御仁の仕事を一手に引き受けて働いていたMZ-2000君では，CPUが自分でKeySenseを操作してKeyDataをチェックし，キー入力の有無を判定するようになっていたのだが，Xシリーズは初代X1の時代よりこの泥臭い作業から足を洗ってしまっている。とはいえ，誰かがキーをスキャンしなければキー入力を得ることなどできやしない。CPUに代わってこの役目を仰せつかっているのがサブCPUである。

いや，サブCPUという言葉の響きには，CPUをサポートしてさまざまな周辺処理をこなし，さらにはプログラマブルであるかのような印象があるな。もうちょっと現実的にいうならば，「かくかく，しかじか，のように動作すべし」とあらかじめ命令された1チップマイコンがこの作業に当たっている，ということになる。吾輩のキーボードに使用されているのは80C51である。こいつは常にキーをスキャンしてキー入力を見張っており，なにかキーが押されたら，「キーが押されたよ」と吾輩のCPUに通知してくるようになっている。そう，キー入力割り込みである。

前回のマウスに続いて再び割り込みが登場したが，いくつものデバイスが処理を分担し，必要に応じて，必要な時間だけ，CPUに割り込みをかけてデータをやり取りするというのは，現在のパソコンの常識になっている。いっさいがっさいの面倒をすべてCPUが見るのでは，CPUの処理時間があまりに無駄に使われてしまう。たとえば，一定の時間ごとにキー入力がないかどうかをCPUがチェックするシステムを考えてみられたい。このシステムではキーが押されていようが押されていまいが，キースキャンに要する時間が一定時間ごとに消費されてしまうことになる。かといって，OSの1文字入力ルーチンや1行入力ルーチンが実行されたときだけキー入力をチェックするというのでは，キー入力の取りこぼしが発生する可能性大である。もちろん，カーソルが点滅する前に次の命令を入力してしまう先行入力など実現できようはずもない。キー入力が発生したら，その旨を割り込みで通知するメリットはここにある。

吾輩のキーボードでは，キー入力割り込みはワンチップマイコンからの「キーデータ転送」によって発生する。ワンチップマイコンは押されたキーのデータを，マウス同様シリアル通信で吾輩に転送してくる。データフォーマットは，2400bps，スタートビット1，ストップビット2，パリティなしのデータ長8ビットである。吾輩のMFP(Multi Function Peripheral)に内蔵されたシリアル通信インタフェイスUSART(Universal Synchronous/Asynchronous Receiver/Transmitter)がこのデータを受け取り，1ビット1ビット届くデータを1バイトにパックして，「データが届いたよ」と吾輩に割り込みをかけてくる。吾輩はこのキー入力割り込みがあるまでは自分の仕事に専念していられるわけである。

割り込みがかかると，吾輩はそれまでの作業を一時中断して転送されてきたキーデータを受け取り，それをメモリに格納しておく。そして，OSやIOCSの1文字入力ルーチンが要求してきたときにそれを取り出して渡してやるのである。諸兄がうっかり，

A>dir

とやってハードディスクのディレクトリを表示してしまい，それが延々と表示されている途中で，

A>dir ＊.bas

と再入力したのを吾輩がちゃんと覚えていて，ディレクトリの表示終了と同時に新しい指示に従ってディレクトリを再表示できるのは，このような仕組が存在するためである(もっとも，すぐさま^Cでディレクトリ表示を中断なさるだろうが)。

◆2つのキーの同時判断

キーボードから送られてくるデータは，個別のキーに割り振られた特殊なコードである。キー入力を1文字1文字処理していくプログラムにはいいのだが，ゲームのように複数のキーを同時に判定したいという場合にはこれでは役に立たない。そこで吾輩のIOCSには，キーボードから送られてきたデータをデコードし直し，あたかもキーマトリクスを直接走査しているような処理ができるルーチンが用意されている。IOCSコールNo.4である。

図5を見ていただきたい。D1.Wに図のグループNo.をセットしてIOCSコールNo.4を利用すると，D0.Bには対応するキーが押されているかどうかが1/0のビット並びとして返されるようになっている。たとえばカーソル右とカーソル上の2つのキーが同時に押されているならば，D1.W=7としてIOCSコールNO.4を利用すると，00011000_BというデータがD0.Bに返ってくるわけである。

今回最後にお届けするのは，これを実際に試してみるプログラムである(図6)。ここでは，グループNo.0の7つのキーの状態を画面に表示している。ESCキーはプログラム終了のキーとして使っているので，1～6のキーを押してビット列の変化を楽しんでみていただきたい。プログラムの最後でIOCSコールNo.2とNo.3を使っているが，これはメモリに蓄えられたキー入力を破棄するためである。他意はない。

ひととおり遊び終えたら，是非とも試してみていただきたいことがある。メインキーの1，TAB，Qのキーを同時に押してみていただきたいのだ。結果は見てのお楽しみ。タネ明かしは次回行う予定である。

図5　グループNo.と対応するキー

グループNo.	D0.bの対応するビット							
	0	1	2	3	4	5	6	7
0		ESC	1	2	3	4	5	6
1	7	8	9	0	-	^	¥	BS
2	TAB	Q	W	E	R	T	Y	U
3	I	O	P	@	[	RET	A	S
4	D	F	G	H	J	K	L	;
5	:	]	Z	X	C	V	B	N
6	M	,		/	_	スペース	HOME	DEL
7	ROLL UP	ROLL DN	UNDO	←	↑	→	↓	CLR
8	(/)	(*)	(-)	(7)	(8)	(9)	(+)	(4)
9	(5)	(6)	(=)	(1)	(2)	(3)	ENTER	(0)
A	(,)	(.)	記号	登録	HELP	XF1	XF2	XF3
B	XF4	XF5	かな	ローマ字	コード入力	CAPS	INS	ひらがな
C	全角	BREAK	COPY	F1	F2	F3	F4	F5
D	F6	F7	F8	F9	F10			
E	SHIFT	CTRL	OPT.1	OPT.2				

図6　2キー同時入力の実験

```
-z0=200000
-an .z0
        †_exit           equ      $ff00

        loop:
00200000          moveq   #0,d1                    * キーコードグループ0をスキャン
00200002          moveq   #$04,d0                  * _bitsns
00200004 .        trap    #15
00200006          btst.l  #1,d0                    * ESCキーが押されたか
        †        bne.s   end                      * そうなら終了
0020000A          bne.s   .z0+$2e

0020000C          move.b  d0,d3                    * キーの状態をD3.bに保存
0020000E          moveq   #7,d2
        prnt:                                      * それを2進数で画面に表示する
00200010          moveq   #'0',d1
00200012          btst.l  d2,d3                    * 第D2ビットは1か
        †        beq.s   prnt1                    * 違うならprnt1へ
00200014          beq.s   .z0+$18
00200016          moveq   #'1',d1
        prnt1:
00200018          moveq   #$20,d0                  * _b_putc
0020001A          trap    #15
        †        dbra    d2,prnt
0020001C          dbra    d2,.z0+$10
00200020          moveq   #$0d,d1                  * 改行を表示
00200022          moveq   #$20,d0
00200024          trap    #15
00200026          moveq   #$0a,d1
00200028          moveq   #$20,d0
0020002A          trap    #15
        †        bra.s   loop                     * 以上の繰り返し
0020002C          bra.s   .z0

        end:
0020002E          moveq   #$02,d0                  * _b_sftsns
00200030          trap    #15
00200032          lsr.w   #8,d0                    * 現在のLEDの状態を
00200034          move.b  d0,d1                    * D1にセット
00200036          moveq   #$03,d0                  * _b_key_init
00200038          trap    #15                      * これで先行入力がクリアされる
        †        dc.w    _exit
0020003A          _exit
```

X68000・
Z-MUSIC+PCM8用

FIRE CRACKER

Mori Hirosi 森 弘

X68000・Z-MUSIC用
(SC-55対応)

サンバDEグワッシャ!!

Shozi Singo
荘司 真吾

年も明けて寒さもいよいよ本番ですね。おウチにこもってパソコンで遊んでばかりの人への贈り物として，運動不足解消に最適な(?)ノリのいい2曲をご紹介しましょう。根強い人気のYMOと，ゲームミュージックのイメージで作ったオリジナル曲です。

炎割り人形

さて，今月の1曲目はYMOのファーストアルバム「YELLOW MAGIC ORCHESTRA」から「FIRE CRACKER」をお届けしましょう。PCM8.Xが必要です。

見出しの「炎割り人形」には深い意味はありません。ただ，チャイコフスキーの「くるみ割り人形」が「NUT CRACKER」だから，和訳するなら炎割り人形でいいかな，なんて考えただけです。

YMOの説明はいらないでしょう。今や日本を代表するアーティストの坂本龍一さんや細野晴臣さん，高橋幸宏さんという豪華メンバーのグループです。現在は散開(＝解散！)していますが，人気の高さは今でも健在といったところでしょうか。

この作品では，リミックス版とでもいうようなアレンジが施されています。オープニングは散開コンサートバージョンのように「裏から入るリズム」になっています。これは高橋幸宏さんの得意パターンで，俗にいう「裏打ち」というやつですね。途中はオリジナルバージョン，エンディングは散開コンサートバージョンになっています。

内蔵音源だけなのでちょっと音の厚さが足りないような気もしますが，音色などはよく似ています。

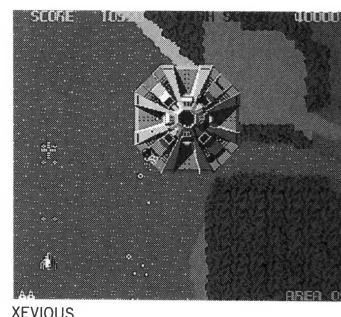
YELLOW MAGIC ORCHESTRA

リストの右端のコメントの部分にアスタリスク“＊”がある行は，上のほうに同じ内容の行があります。入力時の参考にしてみてください。

ゲームイメージより，強烈な個性?

さて，もう1曲もX68000用で，SC-55が必要になります。タイトルは「サンバDEグワッシャ!!」。明るく，軽快な曲です。

これはオリジナル曲なのですが，実はあの有名なゲーム「XEVIOUS」のBGMからイメージをふくらませて作ったということです。よく聴いてみるとイメージのなごりがあり，「なるほど」って感じですが，全体的にはゲームのイメージとはまたひと味違った曲に仕上がっています。

XEVIOUS

サンバが大好きという作者の荘司くんの趣味なんだそうですが，なかなかかっこよくキマっています。私が初めて聴いたときは，あまりのうまさに膝ポンまでしてしまいました。みなさんもぜひ入力して聴いてみてください。きっと膝ポンしちゃうと思いますよ。荘司君はこれ以外にもオリジナル曲を送ってくれましたが，そちらも素晴らしかったですね。CM-500を買ったそうですから，これからも期待してますよ。

このページへ送られてくる作品は，ジャンルはさまざまですが，原曲を忠実に再現したり，それをアレンジしたものがやはり多くを占めています。そういう曲はもちろん大歓迎ですが，このようなユニークで大胆なオリジナル曲もOKです。「オリジナルなんてウケないんじゃないか」なんて思わずにどんどん投稿してね。 (S.K.)

リスト1 FIRE CRACKER


```
 1: /==================================================
 2: /    FIRECRACKER      YMO
 3: /    マーティン・デニー作曲
 4: /==================================================
 5: .ADPCM_BLOCK_DATA=FIRE.ZPD
 6: (I)
 7: (B0)
 8: (M1,4000)(AFM1,1)
 9: (M2,4000)(AFM2,2)
10: (M3,4000)(AFM3,3)
11: (M4,4000)(AFM4,4)
12: (M5,4000)(AFM5,5)
13: (M6,4000)(AFM6,6)
14: (M7,4000)(AFM7,7)
15: (M8,4000)(AFM8,8)
16: (M9,4000)(AADPCM,9)
17: (M10,4000)(AADPCM,10)
18: (M11,4000)(AADPCM,11)
19: (M12,4000)(AADPCM,12)
20: (M13,4000)(AADPCM,13)
21: /==================================================
22: /SOUND
```


(S.K.)

```
23: /==================================================
24: /-------------------
25: /SYNTH 1
26: /-------------------
27: /      AR DR SR RR DL TL KS MT DT DT AM
28: (@1,   29,28,00,00,00,20,00,02,03,00,00
29:        31,31,04,06,00,10,00,01,03,00,00
30:        31,31,00,00,00,26,00,02,07,00,00
31:        31,12,00,06,15,00,00,01,07,00,00
32: /      AL FB
33:        04,04,15,03)
34: /----------------
35: /SYNTH 2 ﾋﾟｯｺﾛ
36: /----------------
37: /      AR DR SR RR DL TL KS MT DT DT AM
38: (@2,   18,00,00,07,00,45,00,02,07,00,00
39:        29,00,00,08,00,00,00,02,00,00,00
40:        18,00,00,07,00,40,00,02,00,00,00
41:        25,00,00,08,00,00,00,02,03,00,00
42: /      AL FB
43:        04,02,15,03)
44: /----------------
45: /SYNTH BASS
46: /----------------
47: /      AR DR SR RR DL TL KS MT DT DT AM
48: (@5,   31,05,05,06,05,25,00,01,07,00,00
49:        18,06,03,06,05,50,00,00,00,00,00
50:        31,05,03,06,03,35,00,00,03,00,00
51:        31,03,03,08,03,00,00,01,00,00,00
52: /      AL FB
53:        01,05,15,03)
54: /----------------
55: /STRINGS
56: /----------------
57: /      AR DR SR RR DL TL KS MT DT DT AM
58: (@12,  31,31,00,00,00,26,00,02,07,00,00
59:        20,06,00,06,03,00,00,02,07,00,00
60:        31,31,00,00,00,28,00,04,03,00,00
61:        20,31,00,06,00,06,00,02,03,00,00
62: /      AL FB
63:        04,07,15,03)
64: /----------------
65: /ﾋﾟｱﾉ
66: /----------------
67: /      AR DR SR RR DL TL KS MT DT DT AM
68: (@13,  31,05,02,01,01,35,01,01,07,00,00
69:        25,05,02,01,01,45,02,13,00,00,00
70:        31,05,02,01,01,38,01,03,03,00,00
71:        20,06,03,03,10,00,02,01,00,00,00
72: /      AL FB
73:        02,06,15,03)
74: /----------------
75: /ﾌｴ
76: /----------------
77: /      AR DR SR RR DL TL KS MT DT DT AM
78: (@20,  10,31,00,00,00,33,00,04,07,00,00
79:        20,31,00,07,00,00,00,02,07,00,00
80:        10,31,00,00,00,33,00,04,03,00,00
81:        20,31,00,07,00,00,00,02,03,00,00
82: /      AL FB
83:        04,00,15,03)
84: /----------------
85: /ｼｰｹﾝｽﾖｳ PSGﾌｳ
86: /----------------
87: /      AR DR SR RR DL TL KS MT DT DT AM
88: (@30,  31,31,00,00,00,35,01,08,00,00,00
89:        31,19,00,07,15,00,01,04,00,00,00
90:        31,31,00,00,00,35,01,04,00,00,00
91:        31,15,00,07,15,00,01,02,00,00,00
92: /      AL FB
93:        04,00,15,03)
94: /----------------
95: /ﾒﾛﾃﾞｨ1
96: /----------------
97: /      AR DR SR RR DL TL KS MT DT DT AM
98: (@40,  31,12,00,00,08,30,00,04,00,00,00
99:        31,10,00,06,15,00,00,02,00,00,00
100:       31,12,00,00,08,30,00,02,00,00,00
101:       31,10,00,06,15,00,00,01,00,00,00
102: /     AL FB
103:       04,00,15,03)
104: /----------------
105: /ﾌﾞﾗｽ
106: /----------------
107: /     AR DR SR RR DL TL KS MT DT DT AM
108: (@41, 31,31,00,00,00,30,00,01,00,00,00
109:       25,31,00,07,00,00,00,01,00,00,00
110:       31,31,00,00,00,25,00,01,00,00,00
111:       15,31,00,07,00,00,00,01,00,00,00
112: /     AL FB
113:       04,07,15,03)
114: /----------------
115: /ﾊﾞｲｵﾘﾝ
116: /----------------
117: /     AR DR SR RR DL TL KS MT DT DT AM
118: (@42, 25,02,00,05,01,33,01,01,00,00,00
119:       28,06,00,08,03,30,01,06,00,00,00
120:       29,03,00,06,01,48,01,01,00,00,00
121:       28,00,00,05,00,00,01,01,00,00,00
122: /     AL FB
123:       02,07,15,03)
124: /==================================================
125: /MML DATA
126: /==================================================
127: /--------------------------------------------------
128: /SYNTH 1 ﾒﾛﾃﾞｨ
```

```
129: /--------------------------------------------------
130: (T1)
131: |:8R1:|@40|:                                          /I
132: O4V13L8|:Q4<DE16D16>BAGEDE L16GAGEL8D>BAB<DEL16 /A
133: |:3DED>B|A8<:| >L8AABB<DDEE G4.G2A B4.B2&B8:|   /A
134: O4V13L4.@12Q8@M8@S8                              /B
135: |:@Z115,117,118,119 B&A8B&A8 |:F#&E8:|:|         /B
136: |:@Z115,117,118,119 >B&A8B&L8A @ZB<D>BAF#EF#A:| /B
137: O4L16@42Q4@M6@S4 V10|:14¯1B<C>:|¯1B<CDD#        /B
138: V10|:14¯1EF:|¯1DD#EE# F#8R2.. R*576              /B
139: O4V13L4.@40Q8@M |:<C>B8:| AG8AG8                 /C
140: F#E8F#E8 DC8DC8 >BA8BL8A B<D>BAB<D>BA L1BR :|    /C
141: |:8 R1 :|@1                                      /D
142: O4V13L8|:Q4<DE16D16>BAGEDE L16GAGEL8D>BAB<DEL16 /A*
143: |:3DED>B|A8<:| >L8AABB<DDEE G4.G2A B4.B2&B8:|   /A*
144: O4V13L4.@12Q8@M8@S8                              /B*
145: |:@Z115,117,118,119 B&A8B&A8 |:F#&E8:|:|         /B*
146: |:@Z115,117,118,119 >B&A8B&L8A @ZB<D>BAF#EF#A:| /B*
147: @MO4L16@1Q4 V10|:14¯1B<C>:|¯1B<CDD#             /B
148: V10|:14¯1EF:|¯1DD#EE# F#8R2.. R*576              /B*
149: O4V13L4.@12Q8@M8@S8 <C>B8<C>B8 AG8AG8            /C
150: F#E8F#E8 DC8DC8 >BA8BL8A B<D>BAB<D>BA L1BR       /C*
151: |:O4V13L8@1@M Q4<DE16D16>BAGEDE                  /L
152: DE16D16>BAGEDE<:| O6V11@42@M6@S4L16>             /L
153: |:|:4GAGE:||:3DED>B<:|>AB<DE:|                   /L
154: |:|:4B<D>BA:||:3GAGE:|DE>BA<:|                   /L
155: |:4B<D>BAGAGEGAGEDEGA:| |:4|:B<D>BAGEDE:|:|      /L
156: |:12<DED>B:|                                     /L
157: /--------------------------------------------------
158: /ﾋﾟｯｺﾛ,ﾋﾟｱﾉ,[C]ﾉ ｳﾞｧｲｵﾘﾝ
159: /--------------------------------------------------
160: (T2)
161: L1RRR O5V10L8Q3@2 R*108D16EDE                    /I
162: |:4GAB8.DD16EDE:|                                /I
163: |:12GAB8.DD16EDE:|                               /A
164: |:12R1:| O2V12@13L16P3Q8                         /B
165: {F#RR<F#RR}*90 {DA#G<G<C>BF#F#GA#GG<CC>}*486     /B
166: O4{GRR<DRR>F#RR>GRR<CRRD>RRF#>R}1                /B
167: O5V12L4.Q8@42@M6@S4 <C>B8<C>B8 AG8AG8 F#E8F#E8  /C
168: DC8DC8 >BA8BL8A B<D>BAB<D>BA B1 <B2A2 @M         /C
169: O5V10L8Q3@2 |:12GAB8.DD16EDE:|                   /A
170: |:12R1:| O4V6@20@M30P1@S22L1Q8 @B-3000,0,0       /B
171: C*576& @B0,8000,0C2.&@B8000,-3500,0C4 @B0@M      /B
172: O5V12L4.Q8@42@M6@S4 <C>B8<C>B8 AG8AG8 F#E8F#E8  /C*
173: DC8DC8 >BA8BL8A B<D>BAB<D>BA B1 <B2A2 @M         /C*
174: |:3R1:| O5V10L16Q3@2 R2RDERL8DE                  /D
175: O5V10L8Q3@2 |:4GAB8.DD16EDE:|                    /D
176: |:12GAB8.DD16EDE:|                               /A*
177: |:12R1:| O5V13@13L4P3Q4 >A#<F#A#R <AEGDQ8        /B

/B
178: L32F#8F#GF#C#>{GF#GF#C#>GF#GF#C#>GF#GF#C#}*144  /B
179: {F#RRARR<CRRD#RRF#RRARR<ER}*192                  /B
180: O5V12L4.Q8@42@M6@S4 <C>B8<C>B8 AG8AG8 F#E8F#E8  /C*
181: DC8DC8 >BA8BL8A B<D>BAB<D>BA B1 <B2A2 @M         /C*
182: |:3R1:| O5V10Q3@2 L16R2RDERL8DE                  /L
183: |:19GAB8.DD16EDE:|                               /L
184: /--------------------------------------------------
185: /SYN 1,ﾌｴ,ﾌﾞﾗｽ,ﾋﾟｱﾉ
186: /--------------------------------------------------
187: (T3)
188: |:8R1:|                                          /I
189: O4V13@40 |:R*768 L4.Q4DQ8D2E8 Q4F#Q8F#*120:|>   /A
190: V9@20L1P2 RRRR |:F# A2E4F#4:|                    /B*
191: L8Q4|:16F#:| |:16G#:|                            /B*
192: O2V12@13L16P3Q8 {RA#RRGR}*90                     /B
193: {R<GRR<CRR>BRR<F#RR<F#RRDRR>A#R}*243             /B
194: {RA#RRBRR<CRR>BRRF#RRF#RRGR}*243                 /B
195: O4{RBRR<CRR>DRRF#RR>GRRCRR>C}1                   /B
196: O5V7@41L4.P2@M6@S5 <AG8AG8 F#E8F#E8              /C
197: DC8DC8 >BA8BA8 L1B A G <F#2E2                    /C
198: |:O4V13L8@40P3@M R*768 DR4D2E F#4.F#2&F#:|>      /A
199: V9@20L1P2 RRRR |:F# A2E4F#4:|                    /B*
200: L8Q4|:16F#:| |:16G#:|                            /B*
201: O4V6@20@M30@S17P2L1 @B-3000,0,0                  /B
202: C*576& @B0,8000,0C2.&@B8000,-3500,0C4 @B0        /B
203: O5V7@41L4.P2@M6@S5 <AG8AG8 F#E8F#E8              /C*
204: DC8DC8 >BA8BA8 L1B A G <F#2E2                    /C*
205: |:8R1:|                                          /D
206: O4V13@1P3@M |:R*768 L4.Q4DQ8D2E8 Q4F#Q8F#*120:|>/A
207: V9@20L1P2 RRRR |:F# A2E4F#4:|                    /B*
208: L8Q4|:16F#:| |:16G#:|                            /B*
209: O5V13@13L4P3Q4 R*3C#A#<DR E>B<D>AQ8 R*189        /B
210: O3{RGRRA#R<RC#RRERRGRRA#R<RE#}*192               /B
211: O5V7@41L4.P2@M6@S5 <AG8AG8 F#E8F#E8              /C*
212: DC8DC8 >BA8BA8 L1B A G <F#2E2                    /C*
213: O4V10L8@20@M8@S6 |:|:<DE16D16>BAR2>:|<:|@M       /L
214: L16V9 |:|:4GAGE:||:3DED>B<:|>AB<DE:|             /L
215: |:|:4GAGE:||:3DED>B<:|>AB<DE:|                   /L
216: |:8|:GAGE:||:DED>B<:|:| |:3|:4GAGE:|:|           /L
217: /--------------------------------------------------
218: /ｳﾞｧｲｵﾘﾝ,ﾌﾞﾗｽ,ﾋﾟｱﾉ
219: /--------------------------------------------------
220: (T4)
221: |:8R1:|                                          /I
222: O5V11@42L8P2@M6@S4                               /A
223: |:Q4|:4GR4GR2:| @K5GR4Q8G2A Q4BR4Q8B2&B:|@K0     /A
224: |:12R1:| O2V12@13L16P3@M                         /B
225: {RR<CRR>B}*90{RR<F#RR<F#RRDRRA#RRGRRGRR<C}*243   /B
226: {RRCRR>F#RR<F#RRDRR>A#RRA#RRB}*243               /B
227: O4{RR<F#RR>GRRCRRDRR>F#RR>GRR}1                  /B
228: O5V10@41L4.@M6@S5 |:C>B8<C>B8:| AG8AG8           /C
229: F#E8F#E8 F#E8F#L8E |:F#AF#D:| E1 F#2R2           /C
230: O5V11@42L8P2@M6@S4                               /A*
231: |:Q4|:4GR4GR2:| @K5GR4Q8G2A B4RB2&B:|@K0         /A
232: |:16R1:|                                         /B
```

```
233: O5V10@41L4.@M6@S5 |:C>B8<C>B8:| AG8AG8                /C*
234: F#E8F#E8 F#E8F#L8E |:F#AF#D:| E1 F#2R2                /C*
235: |:8R1:|                                                 /D
236: O5V11@42L8P2@M6@S4                                      /A*
237: |:Q4|:4GR4GR2:| @K5GR4Q8G2A Q4BR4Q8B2&B:|@K0           /A*
238: |:12R1:| O5V13@13L4P3Q4 R*6F#<DF#R C>G#B-F#Q8          /B
239: R*186 O3{RRG#RRB<RRDRRE#RRG#RRB<RR}*192                 /B
240: O5V10@41L4.@M6@S5 |:C>B8<C>B8:| AG8AG8                /C*
241: F#E8F#E8 F#E8F#L8E |:F#AF#D:| E1 F#2R2                /C*
242: |:12R1:| O5V11@42L8P2@M6@S4 Q4|:11GR4GR2:|            /L
243: /---------------------------------------------------------
244: /ｳﾞｧｲｵﾘﾝ,ﾋﾟｱﾉ
245: /---------------------------------------------------------
246: (T5)
247: |:8R1:|                                                 /I
248: O6V9@42L8P1@M6@S4                                       /A
249: |:Q4|:4GR4GR2:| GR4Q7G2G Q4GR4Q7G2&G:|Q8               /A
250: L1RRRR |:RF#2.E4:| L16|:16RF#:| |:16RG#:| R*768 /B
251: L4.<C>B8<C>B8 AG8AG8 F#E8F#E8 DC8DC8                    /C
252: >BA8BL8A B<D>BAB<D>BA L2B<EBA                           /C
253: O6V9@42L8P1@M6@S4                                       /A*
254: |:Q4|:4GR4GR2:| GR4Q7G2G G4RG2&G:|Q8                   /A
255: L1RRRR |:RF#2.E4:| L16|:16RF#:| |:16RG#:| R*768 /B*
256: L4.<C>B8<C>B8 AG8AG8 F#E8F#E8 DC8DC8                    /C*
257: >BA8BL8A B<D>BAB<D>BA L2B<EBA                           /C*
258: |:8R1:|                                                 /D
259: O6V9@42L8P1@M6@S4                                       /A*
260: |:Q4|:4GR4GR2:| GR4Q7G2G Q4GR4Q7G2&G:|Q8               /A*
261: L1RRRR |:RF#2.E4:| L16|:16RF#:| |:16RG#:|              /B
                                                       /[B]
262: O6V10@13L32P3@M R1 R1 R*9 F#8F#GF#C#>                   /B
263: {GF#GF#C#>GF#GF#C#>GF#GF#C#}*144 R*183                  /B
264: O6V9@42L8P1@M6@S4                                       /C*
265: L4.<C>B8<C>B8 AG8AG8 F#E8F#E8 DC8DC8                    /C*
266: >BA8BL8A B<D>BAB<D>BA L2B<EBA                           /C*
267: |:12R1:| O6L8Q4|:11GR4GR2:|                            /L
268: /---------------------------------------------------------
269: /ｼｰｹﾝｽ (RIGHT)
270: /---------------------------------------------------------
271: (T6)
272: L1RRRR O2V13Q4L8@30P2 |:8G<G>R16G<G16>:|               /I
273: |:3 |:24G<G>R16G<G16>:|                                 /A
274: |:8B<B>R16B<B16>:| |:|:B<B>R16B<B16>:|                 /B
275: <D<D>R16D<D16>E<E>R16F#<F#16>>:|                        /B
276: |:4B<B>R16B<B16>:| <|:4E<E>R16E<E16>:|                 /B
277: |:F#<F#>R16F#<F#16>:|< |:F#<F#>R16F#<F#16>:|<          /B
278: |:4F#<F#>R16F#<F#16>:|>>                                /B
279: |:8C<C>R16C<C16>:|> |:B<B>R16B<B16>:|                  /C
280: |:G<G>R16G<G16>:| B<B>R16B<B16>                         /C
281: < C#<C#>R16C#<C#16> |:D<D>R16D<D16>:|> |1:|            /C
282: |2 R*768 |:8G<G>R16G<G16>:| :|                          /D
283: |3 |:8G<G>R16G<G16>:| |:38G<G>R16G<G16>:|              /L
284: /---------------------------------------------------------
285: /ｼｰｹﾝｽ (LEFT)
286: /---------------------------------------------------------
287: (T7)
288: L1RRRR O2V13Q4L8@30P1 |:8G16<G>G8.G:|                  /I
289: |:3 |:24G16<G>G8.G:|                                    /A
290: |:8B16<B>B8.B:| |:|:B16<B>B8.B:|                       /B
291: <D16<D>D8.DF#16<F#>F#8.F#> :|                           /B
292: |:4B16<B>B8.B:| <|:4E16<E>E8.E:|                       /B
293: |:F#16<F#>F#8.F#:|< |:F#16<F#>F#8.F#:| <               /B
294: |:4F#16<F#>F#8.F#:|>>                                   /B
295: |:8C16<C>C8.C:|> |:B16<B>B8.B:| |:G16<G>G8.G:|        /C
296: B16<B>B8.B< C#16<C#>C#8.C# |:D16<D>D8.D:|> |1:| /C
297: |2 R*768 |:8G16<G>G8.G:| :|                             /D
298: |3 |:8G16<G>G8.G:| |:38G16<G>G8.G:|                    /L
299: /---------------------------------------------------------
300: /ﾍﾞｰｽ
301: /---------------------------------------------------------
302: (T8)
303: O3V11Q3@5 L1RRRL8 R2R16D16EDE |:4GAB8.DD16EDE:| /I
304: |:3 |:12GAB8.DD16EDE:|                                  /A
305: |:4B<CD8.>F#F#16GF#A:|                                  /B
306: |:B<CD8.>F#F#16GF#A <DEF#8.>F#F#16F#GA:|                /B
307: B<CD#8.>F#F#16GF#A B<CD#8.>BB16<C>B<C>                  /B
308: |:<EFG#8.>BB16<C>B<D>:| R*768                           /B
309: |:4<CDE8.>GG16AGA:| B<C#D8.>F#F#16GF#A                  /C
310: GAB8.DD16EDE CDE8.>GG16AGA <DEF#8.>AA16BAB |1:| /C
311: |2 L1RRRR O3V11Q3L8@5 |:4GAB8.DD16EDE:| :|             /D
312: |3 L1RRR O3L16 R2RDERL8DE |:19GAB8.DD16EDE:|           /L
313: /---------------------------------------------------------
314: /PERCUSSION(ｴｽﾆｯｸ)
315: /---------------------------------------------------------
316: (T9)
317: O4L8 |:200RBRC16C16:| |:94RBRC16C16:|
318: /---------------------------------------------------------
319: /SNARE,BOMB
320: /---------------------------------------------------------
321: (T10)
322: O3L4 R*576 R2R16C16C8C8C8 |:108RCRC:|
323: R*768 |:31RCRC:| RO5C*768
324: /---------------------------------------------------------
325: /B.D,HANDCLAP
326: /---------------------------------------------------------
327: (T11)
328: L1RRRRL4 |:108O3BRBO7C:| |:4R2.O7C:| |:31O3BRBO7C:|
329: /---------------------------------------------------------
330: /ﾍﾞﾝﾄﾞｽﾙ B.D
331: /---------------------------------------------------------
332: (T12)
333: O6L1 RRRR |:143C:|
334: /---------------------------------------------------------
335: /SHAKER
336: /---------------------------------------------------------
337: (T13)
338: O3L1 RRRR |:143R*35A*26A*35R*35A*26A*35:|
339: /=========================================================
340: /PLAY
341: /=========================================================
342: (O122)
343: (P)
```

リスト2　FIRE CRACKERの音色コンフィグファイル

```
.o3c=sd808.pcm
.o3a=shaker.pcm
.o3b=drk2.pcm
.o4c=wblk.pcm,v40
.o4b=wblk.pcm,p11,v40
.o5c=bomb1.pcm
.O6c=etomg.pcm,p-12,v50
.O7d=clapm1.pcm
.O7c=snap.pcm,v50,mo7d
```

リスト3　FIRE CRACKERのカウンタ表示

```
 1:00006E40 00000000   2:00006E40 00000000   3:00006E40 00000000   4:00006E40 00000000
 5:00006E40 00000000   6:00006E40 00000000   7:00006E40 00000000   8:00006E40 00000000
 9:00006E40 00000000  10:00007170 00000000  11:00006E40 00000000  12:00006E40 00000000
13:00006E40 00000000
```

リスト4　サンバDEグワッシャ!!

```
 1: .comment - サンバ DE グワッシャ!! -  Programmed by しょうじ しんご
 2:
 3: /---------------------------------------------------------
 4: / TRACK SETUP
 5:
 6: (i)
 7: (b1)              / Base Channel = MIDI
 8:
 9: (m10,3000)(aMidi10,10)
10: (m11,3000)(aMidi10,11)
11: (m12,3000)(aMidi10,12)
12: (m13,3000)(aMidi10,13)
13: (m1 ,3000)(aMidi1 , 1)
14: (m2 ,3000)(aMidi2 , 2)
15: (m3 ,3000)(aMidi3 , 3)
16: (m4 ,3000)(aMidi4 , 4)
17: (m5 ,3000)(aMidi5 , 5)
18: (m6 ,3000)(aMidi6 , 6)
19: (m7 ,3000)(aMidi7 , 7)
20:
21: /---------------------------------------------------------
22: / SC55 INIT
23:
24: .roland_exclusive 16,66={$40,00,s7F,00}
25:
26: /---------------------------------------------------------
27: /VOICE RESERVE
28:
29: .sc55_v_reserve $10={2,3,2,2, 3,2,3,0 ,0,7,0,0 ,0,0,0,0}
30:
31: /---------------------------------------------------------
32: / MML DATA SET
33:
34: /t2とt5は似ています。 また、t3とt4も似ています。
35:
36: (t1) @i$41,$10,$42 @e90,50 @m @q0@g12@b0@p90@57@v90 @u12
0 o5@k0 r418
37: (t1) r4c1'ce1''cg1'<c1>
38:
39: (t1) ~20l16|:frrfrrcr_25f~25ar<c>arfr|grgrgrd8.grgfrer:|
grgrgr<c4rr(c4f),24r1r1r1r1r64_10
40:
41: (t1) @e120,50>@p0|:frfrfrcfrarb+arfrgrgrgr|gdrgrgfrcr>a4
```

▶ハードディスクと増設RAMは，絶対一緒にほしい。どっちかだけ買うというのは，なんか無駄な気がします。そう思いません？
半田　文夫(18)長野県

```
.<c8^2&r1@p108~20:|ab-rb-rb-argrf1&r1
    42:
    43: (t1) _10@e90,50@p64l16cr4..c8.d8.c8>b-r4..b-8.a8.b-8<cr4
..c8.c8.c8c2>b2
    44:
    45: (t1) @58 @h70@m90 @e120,100 |:r1r1r1|r1:|r2.@v120
    46: (t1) (c-4c)&r2l12(c6d)>b-ab-(g2a)&rfgagf(e4c)&r^4c8.d8.e
4f4.fgefgag2b-8.a8.g8(a2<c),48&r2
    47: (t1) >g2.< @m y0,8@49<_15l16c4frfrerfry0,0@57@e90,50@v90
o5r4
    48:
    49: (t1) r4c1'ce1''cg1''c1<c'
    50:
    51: (t1) ~20l16|:frrfrrcr_25f~25ar<c>arfr|grgrgrd8.grgfrer:|
grgrgrfrr2
    52:
    53: (t1)         |:frrfrrcr_25f~25ar<c>arfr|grgrgrdrrgrgfrer:|
_5l8<|:crcrcrcr:|~5d1e1
    54:
    55: (t1) ~5l4rrr(ce),1f8r8>f8r8
    56: /----------------------------------------------------------
    57: (t2) @i$41,$10,$42 @e120,15 @q0@g12@b0@p64@49@v95 @u120
o5@k0 r4l8
    58: (t2) r4c1~1'ce1'~2'c1e'~2'c1<c'
    59:
    60: (t2) l16_5|:'cfa'rr>'a<cf'rr'fa<c'rr<'dfb-'r'cfa''ceg'r'
df>b-'r|
    61: (t2) 'cdg'r'cdg'r'cdg'r'd8>bg'r'gd>b'r'gd>b''fc>a'r'ec>g
'r:|
    62: (t2) 'cfg'r'cfg'r'cfg'r'c4.eg''e4g<c'l2>'a<cf'|:'fb-<d''
g<ce'|'fb-<d''fa<c':|'f8b-<d'r4.r64
    63:
    64: (t2) |:3'fa<c''fb-<d''g<ce'|'fb-<d':|'b-<df''a<cf''fb-<d
''g<ce''fb-<d'
    65:
    66: (t2) <l8'eg<c'r4.'e.g<c''f.b-<d''eg<c''dfb-'r4.'d.fb-''c
.fa''dfb-'
    67: (t2) 'eg<c'r4.'e.g<c''e.g<c''eg<c''g2<cd''g2b<d'>
    68:
    69: (t2) l1_10|:'ceg''c2ga'&'c2fa''cgb-' 'cfa':|
    70: (t2)      <|:'ceg''c2ga'&'c2fa'|'cgb-''cfa':|l16~10'c2gb-
''e2g<c''a<cf'r'a<cf'r'g<ce'r'a<cf'rr4
    71:
    72: (t2) @v95o5r4c1~1'ce1'~2'c1e'~2'c1<c'
    73:
    74: (t2) l16_5|:'cfa'rr>'a<cf'rr'fa<c'rr<'dfb-'r'cfa''ceg'r'
df>b-'r|
    75: (t2) 'cdg'r'cdg'r'cdg'r'd8>bg'r'gd>b'r'gd>b''fc>a'r'ec>g
'r:|
    76: (t2) 'ceg'r'ceg'r'ceg'r'fc>a'rr'c>a''d>b-''ce''df''d+f+'
'eg''eg+'
    77:
    78: (t2)         |:'cfa'rr>'a<cf'rr'fa<c'rr<'dfb-'r'cfa''ceg'r'
df>b-'r|
    79: (t2) 'cfb-'r'cfb-'r'cfb-'r'eg<c'rr'eg<c'r'dfb-''cfa'r'ce
g'r:|
    80: (t2) l4'ceg'~1'cfa'~1'dfb-'~1'eg<c'~1'dfb-'~1'eg<c'~1'fb
-<d'~1'g<ce'~1'f1b-<d''g1<ce'
    81:
    82: (t2) r2.~5'eg<c''a4<cf'>'a8<cf'r8
    83: /----------------------------------------------------------
    84: (t3) @i$41,$10,$42 @e50,50  @q0@g12@b0@p34@38@v97 @u120
o2@k3 r4l16
    85: (t3) fedd-l8@q3|:4~5ccc8.ccc16c4:|
    86:
    87: (t3) l8_15>|:fr16a.<c>b-r16<d.f|cr16e.gfr16>b-.g:|<@q5cc
e@q0g4.c4
    88: (t3) >|:3ffff16fff16ff:|l16frfrff8.f8r4.r64
    89:
    90: (t3) l8|:4ffff16fff16ffffff16fff16|ff:|l16~10fedd-
    91:
    92: (t3) l8|:3cr4.c.c.c:|ggggl16ggf8e8dd
    93:
    94: (t3) l8|:cccc16cc16cdeffff16ff16agfgggg16g16|<d.c.>b-aaa
a16aa16b-b+c16c16:|b-.a.gffff16ff16fed
    95: (t3)    |:cccc16cc16cdeffff16ff16agfgggg16g16|<d.c.>b-aaa
a16aa16b-b+c16c16:|
    96: (t3) b-.a.gl16frfrerfrr4
    97:
    98: (t3) o2fedd-l8@q3@v105|:4~5ccc8.ccc16c4:|
    99:
   100: (t3) l8_15>|:fr16a.<c>b-r16<d.f|cr16e.gfr16>b-.g:|<@q5cc
c>f<@q0r4c4>
   101:
   102: (t3)         |:fr16a.<c>b-r16<d.f|cr16e.gfr16>b-.g:|
   103: (t3) <|:c{cccc}4c{cccc}4cr:||:ccc.ccc16c4:|>
   104:
   105: (t3) l8r2.{cd-de}4f4f4
   106: /----------------------------------------------------------
   107: (t4) @i$41,$10,$42 @e50,10  @q0@g12@b0@p74@59@v87 @u120
o2@k-3r4l16
   108: (t4) fedd-~5c1~5c1~5c1~5c1l8
   109:
   110: (t4) l8_5|:f.a.<c>b-.<d.f|c.e.gf.>b-.g:|<cceg4.c4
   111: (t4) >|:3ffff16fff16ff:|fff16f.fr4.r64
   112:
   113: (t4) l8|:4ffff16fff16ffffff16fff16|ff:|l16~10fedd-
   114:
   115: (t4) l8|:3cr4.c.c.c:|ggggl16ggf8e8dd
   116:
   117: (t4) l8|:cccc16cc16cdeffff16ff16agfgggg16g16|<d.c.>b-aaa
a16aa16b-b+c16c16:|b-.a.gffff16ff16fed
   118: (t4)    |:cccc16cc16cdeffff16ff16agfgggg16g16|<d.c.>b-aaa
a16aa16b-b+c16c16:|
   119: (t4) b-.a.gl16frfrerfrr4
   120:
   121: (t4) o2fedd-@v90~5c1~5c1~5c1~5c1l8
   122:
   123: (t4) l8_5|:f.a.<c>b-.<d.f|c.e.gf.>b-.g:|<@q5ccc>f<@q0r4c
4>
   124:
   125: (t4)      |:f.a.<c>b-.<d.f|c.e.gf.>b-.g:|
   126: (t4) <|:c{cccc}4c{cccc}4cr:||:ccc.ccc16c4:|>
   127:
   128: (t4) l8r2.{cd-de}4f4f4
   129: /----------------------------------------------------------
   130: (t5) @i$41,$10,$42 @e120,30 @q0@g12@b0@p99@62@v100@u120
o4@k0 r4l1
   131: (t5) r4c~3'ce'~3'ceg'~4'c2e<c'&~10'c2e<c'
   132:
   133: (t5) l16_15<|:'cfa'rr>'a<cf'rr'fa<c'rr<'dfb-'r'cfa''ceg'
r'df>b-'r|
   134: (t5) 'cdg'r'cdg'r'cdg'r'd8>bg'r'gd>b'r'gd>b''fc>a'r'ec>g
'r:|
   135: (t5) 'cfg'r'cfg'r'cfg'r'c4.eg''e4g<c''a8<cf'r2..r1r1r1r6
4
   136:
   137: (t5) >_15|:4rr'fa<c''fa<c'r'fa<c'rr'fb-<d''fb-<d'r'fb-<d
'r4.
   138: (t5) 'g<ce''g<ce'r'g<ce'rr|'fb-<d''fb-<d'r'fb-<d'r4:|'b-
<df''b-<df'r'b-<df'r4
   139:
   140: (t5) ~15l8<'eg<c'r4.'e.g<c''f.b-<d''eg<c''dfb-'r4.'d.fb-
''c.fa''dfb-'
   141: (t5) 'eg<c'r4.'e.g<c''e.g<c''eg<c''g2<cd''g2b<d'
   142:
   143: (t5) l1|:rrrr:|_30
   144: (t5) |:'ceg''c2ga'&'c2fa'|'cgb-''cfa':|'c2gb-'l16~10'e2g
<c''a<cf'r'a<cf'r'g<ce'r'a<cf'rr4
   145:
   146: (t5) @v100o4l1r4c~3'ce'~3'ceg'~4'c2e<c'&~10'c2e<c'
   147:
   148: (t5) l16_5<|:'cfa'rr>'a<cf'rr'fa<c'rr<'dfb-'r'cfa''ceg'r
'df>b-'r|
   149: (t5) 'cdg'r'cdg'r'cdg'r'd8>bg'r'gd>b'r'gd>b''fc>a'r'ec>g
'r:|
   150: (t5) 'ceg'r'ceg'r'ceg'r'fc>a'rr'c>a''d>b-''ce''df''d+f+'
'eg''eg+'
   151:
   152: (t5)         |:'cfa'rr>'a<cf'rr'fa<c'rr<'dfb-'r'cfa''ceg'r
'df>b-'r|
   153: (t5) 'cfb-'r'cfb-'r'cfb-'r'eg<c'rr'eg<c'r'dfb-''cfa'r'ce
g'r:|
   154: (t5) @q5l4'ceg'~1'cfa'~1'dfb-'~1'eg<c'~1'dfb-'~1'eg<c'~1
'fb-<d'~1'g<ce'@q0~1'f1b-<d''g1<ce'
   155:
   156: (t5) r2.~5'e4g<c''a8<cf'>r8'a8<cf'r8
   157: /----------------------------------------------------------
   158: (t6) @i$41,$10,$42 @e30,20  @q0@g12@b0@p64@31@v60 @u120
o3@k0 r4l16
   159: (t6) (f4c)~20c1~5'c1e'~10'c1g'~10'c4<c'<~10(c4.<c)&r4.
   160:
   161: (t6) >>l8_30|:fr16a.<c>b-r16<d.f|cr16e.gfr16>b-.g:|<l16c
rcrerg4.(c4f)
   162: (t6) _5l8|:ffff16ddd16dd|eeee16ddd16dd:|l16ereree8.d8r4.
r64
   163:
   164: (t6) l8|:4cc>a<c16ddd16>a<deece16ddd16|>a<d:|~10(f4c)
   165:
   166: (t6) >_5l8cr4.c.e.g>b-r4.b-.<d.fcr4.c.c-.cc4.(dc)>b4.~30
d16d16
   167:
   168: (t6) |:cccc16cc16cdeffff16ff16agfgggg16g16|<d.c.>b-aaaa1
6aa16b-b+c16c16:|b-.a.gffff16ff16fed
   169: (t6) |:cccc16cc16cdeffff16ff16agfgggg16g16|<d.c.>b-aaaa1
6aa16b-b+c16c16:|
   170: (t6) b-4<<<(c4e)_20l16>f8f8e8ffr4o3
   171:
   172: (t6) @v60(f4c)~20c1~5'c1e'~10'c1g'~10'c4<c'<~10(c4.<c)&r
4.
   173:
   174: (t6) >>l8_30|:fr16a.<c>b-r16<d.f|cr16e.gfr16>b-.g:|<@q5c
def@q0r4e4>
   175:
   176: (t6)         |:fr16a.<c>b-r16<d.f|cr16e.gfr16>b-.g:|
   177: (t6) @q5<l4c~1c>~1b-<~1c~1c~1d~1e~1g@q0l8~1ddd.ddd16d4ee
e.eee16e4
   178:
   179: (t6) @p0@5@v90@e120,30l16o5c<rc-rerc-r>c@v120@56o5rrr(c4
e)f4>f4
   180: /----------------------------------------------------------
   181: (t7) @i$41,$10,$42 @e90,30  @q0@g12@b0@p64@5 @v70 @u120
o5@k0 r4l16
   182: (t7) r4|:7c<cc-cecc-c>:|@e30,90@v55@48o2l32|:16c~4:|
   183:
   184: (t7) @e30,30@v120l16f8._35|:f8.f8.f8ff8f8c8c8|c8c8.c8cc8
c8f8.:|c8c8@e30,90_25|:16c32~4:|f4
   185: (t7) @5@e90,30@v80o6l16|:7ecc-c>c|<cc-c:|rrrr4r64
   186:
   187: (t7) @p117@15o5~35l2r1r1cded&r1r1@p0cdef
   188:
   189: (t7) @5@v90l16|:@p110c<cc-cecc-cr2>|>@p0b-<b-ab-<d>b-ab-
r2:|
   190: (t7) @p110l12'dg'@p74'c-d'>@p37'bg'@p0'gd'@p37'bg'<@p74'
c-d'@p110'd2g'
   191:
   192: (t7) @p64l16_15|:r1r1r1r1:||:13c<cc-cecc-c>:|@48@v40o2@e
30,90|:16c32~5:|f8@e30,30c8>a8frr4
   193:
   194: (t7) @5@v70o5r4|:7c<cc-cecc-c>:|@e30,90@v55@48o2l32|:16c
~4:|
   195:
```

▶最近，周囲にMOが普及しているらしい。そういった人間に会うたび，鬼のように増えているデータの量が，ハードディスクすらつけていない私には恐怖すら感じます。

土尾　文紀(20)神奈川県

```
   196: (t7) @e30,30@v120l16f8._35|:f8.f8.f8ff8f8c8c8|c8c8.c8cc8
c8f8.:|c8c8r4@v120c4
   197:
   198: (t7)                   f8._35|:f8.f8|rf8ff8f8c8c8c8c8.c8cc8
c8f8.:|@e30,90_25|:16c32~4:|
   199: (t7) @q5|:c4~1c4~1c4~1c4~1:|@q0_3518|:ccc.ccc16c4:|
   200:
   201: (t7) @p117@5@v90@e120,30l16o5r<crcrcrc>r@v120@56o4rrr{cd
-de}4f4>f4
   202:
   203: /-----------------------------------------------------------
   204: (t10) @i$41,$10,$42 @e50,40 T137 @r1 o2@v100@u120r4l4
   205: (t10) r4|:4cc|cc:|l16drdr<c>afr
   206:
   207: (t10) <c+>|:3rcrdrrcrcrcdrcr|c:|drdrdrdrc<crcc>afr<c+>
   208: (t10) |:3rcrdrccrcrcdrrr|c:|'cd'r'cd'r'cd''cd'rr'cd'rrr6
4afrfr
   209:
   210: (t10) 'c<c+'|:8rcrdrccrcrcdrrr|c:|
   211:
   212: (t10) l8|:3'd<c+'r4.'d.c''d.c''dc':|'dc<c+''dc''dc'l16dd
dd'd8c''d8c''d8c'
   213:
   214: (t10) l8'c<c+'|:8rdrrcdr|crdrcddrc:|<'dc+'>'db''dg''c<c+
'l16r64c8.bdrgl6..
   215:
   216: (t10) l4'c<c+'|:4c|ccc:|l16drdr<c>afr
   217:
   218: (t10) <c+>|:3rcrdrrcrcrcdrcr|c:|drdrdrdrrrrdbrgr
   219:
   220: (t10) <c+>|:3rcrdrrcrcrcdrcr|c:||:'d4c<c+''d4c'|'d4c''d4
c':|'d8.c''dc'br'dgc'd
   221: (t10) ~5'c<c+'|:rcrdrrcrcrcdrcr|c:|
   222:
   223: (t10) r2.~5dddd4'd4<c+'
   224: /-----------------------------------------------------------
   225: (t11)                              @r1 o2          r4l16
   226: (t11) r4|:7f+f+f+f+f+f+f+f+:|<<l32r64|:4bbb|b:|r64
   227:
   228: (t11) o3l16r|:12f+f+f+|f+:|f+rf+rf+rf+4.f+4
   229: (t11) r|:6f+f+f+f+f+f+f+|f+:|f+rf+rf+f+rrf+rrrr4r64
   230:
   231: (t11) r|:16f+f+f+f+f+f+f+|f+:|
   232:
   233: (t11) l8|:3f+r4.f+.f+.f+:|r2l16f+f+f+8f+8f+8
   234:
   235: (t11) |:r1r1r1r1:||:14f+f+f+f+f+f+f+f+:|f+rf+rf+rf+rr4
   236:
   237: (t11) r4>|:7f+f+f+f+f+f+f+f+:|<<l32r64|:4bbb|b:|r64
   238:
   239: (t11) o3l16r|:12f+f+f+|f+:|f+rf+rf+rf+rr2
   240:
   241: (t11) o3l16r|:12f+f+f+|f+:||:4f+rf+rf+rf+r:||:8f+f+f+f+:
|
   242:
   243: (t11) r2.l4f+f+f+
   244: /-----------------------------------------------------------
   245: (t12)                              @r1 o4          r4l16
   246: (t12) r4|:8dddde-e-e-e-:|
   247:
   248: (t12)   r|:6ddde-e-e-e-|d:|drdrdrd4.e-4
   249: (t12)   r|:6ddde-e-e-e-|d:|drdrddrrdrrrr4r64
   250:
   251: (t12)   |:16dddde-e-e-e-:|
   252:
   253: (t12) |:dr4..drrdrrdr|e-r4..e-rre-rre-r:|drdrdrdde-e-e-r
e-re-r
   254:
   255: (t12) |:15dddde-e-e-e-dddde-e-e-e-:|drdre-rdrr4
   256:
   257: (t12) r4|:8dddde-e-e-e-:|
   258:
   259: (t12)   r|:6ddde-e-e-e-|d:|drdrdrdrr2
   260:
   261: (t12)   r|:6ddde-e-e-e-|d:||:8dddde-e-e-e-:|
   262:
   263: (t12) r2.{dddd}4e-4e-4
   264: /-----------------------------------------------------------
   265: (t13)                              @r1 o4          r4l16
   266: (t13) r4|:4grgra-ra-grgrga-a-a-r:|
   267:
   268: (t13) |:grgra-ra-grgrggrgr|a-ra-ra-rrgrgrga-a-a-r:|a-ra-
ra-ra-rr4a-4
   269: (t13) |:3grgra-ra-grgrga-a-a-r:|grgrggrrgrrrr4r64
   270:
   271: (t13) |:8grgra-ra-grgrga-a-a-r:|
   272:
   273: (t13) |:3gr4..grra-rrgr:|grgra-ra-rgggra-ra-r
   274:
   275: (t13) >>l4|:8rd+rd+|rd+r8d+8d+:|l8d+d+d+d+r4
   276:
   277: (t13) <<r4l16|:4grgra-ra-grgrga-a-a-r:|
   278:
   279: (t13) |:grgra-ra-grgrggrgr|a-ra-ra-rrgrgrga-a-a-r:|a-ra-
ra-ra-rrrrggrgr
   280:
   281: (t13) |:grgra-ra-grgrggrgr|a-ra-ra-rrgrgrga-a-a-r:|
   282: (t13) |:2grgra-ra-grgrga-a-a-r:|
   283:
   284: (t13) r2.l4ga-a-
   285: /-----------------------------------------------------------
   286: (p)
```

リスト5　サンバDEグワッシャ!!のカウンタ表示

```
10:00002AC3 00000000  11:00002AC3 00000000  12:00002AC3 00000000  13:00002943 00000000
 1:00002AC3 00000000   2:00002AC3 00000000   3:00002AC3 00000000   4:00002AC3 00000000
 5:00002AC3 00000000   6:00002AC3 00000000   7:00002AC3 00000000
```

……………………(善)のゲームミュージックでバビンチョ……………………

●ベア・ナックルⅡ　古代祐三　CD:ALCA-443
アルファレコード　2,000円(税込)　1/21発売

前作に引き続き，古代祐三氏が音楽を担当。前作同様にタメのここちよいラップ/ハウス系のリズムにコシロ節が展開する。あいかわらずFM音源，PSGの使い方が超人的なうまさ。FM音源でアナログパルスウェーブのような音を出していたり，PSGでTR808風のオープンハイハットを再現してみたりと，音源の制限をものともしない構成力はおみごと。ただし，ゲームの演出を目的とした曲が多いためゲームを知らないと聴きづらい面もある(と感じた)。

お勧め度　8

● MIDI POWER Ver2.0 －X68000 COLLECTION－　CD:KICA-7611
キングレコード　2,800円(税込)　1/21発売

DTMのお手本的アルバム。「グラディウス」「沙羅曼蛇」「A-JAX」がSC-55サウンドで蘇る。前アルバム「Ver1.0」ではX68000版のゲームサウンドが収録されていたが，今回の「Ver2.0」にはこのアルバムのために新たにシーケンスされたものが収録されている。単なる音源のグレードを上げたアレンジではなく，大胆な切り口でそれぞれの曲がリメークされており，聴くものの意表を突いてくれる。「グラディウス」の「モアイ面のテーマ」がラテン調にアレンジされているのには驚いた。

お勧め度　8

●ワールド・ヒーローズ/SNK　CD:PCCB-00106
ポニーキャニオン　1,500円(税込)　1/21発売

ネオジオの作品とあってPCM音をふんだんに駆使したユニークなBGMはなかなか聴きどころがある。最近この手のゲームが多いので，BGMのほうも同じような路線が多くなりがち。聴き手は少し食傷気味というのが正直な感想。演奏時間51:32で1,500円というのはある意味ではお買い得か。

お勧め度　7

●富士山バスター/カネコ　CD:PCCB-00107
ポニーキャニオン　1,500円(税込)　1/21発売

ありがちな日本音階BGMばかりか，と思いきや，意外にもアメリカ刑事ものTVドラマを彷沸させるメロディアスでスマートな曲が多い。ただ，収録のクオリティはノイジーで非常によくない。せっかくの曲がこもった感じに聴こえてしまうのは非常に残念。いまどきこんなのありか，というくらい収録状態がよくない。

お勧め度　8

●熱唱ストリートファイターⅡ CD:PCCB-00103
ポニーキャニオン　3,000円(税込)　発売中

なんとあのストリートファイターⅡのBGMが全曲ボーカルアレンジされて帰ってきた。ブランカのテーマはなんとあの変態奇術集団・電撃ネットワークが歌っているし，E.本田のテーマはウガンダ，ベガのテーマはストロング金剛，そしてザンギエフのテーマは吉本パチパチパンチの島木譲二が歌うという始末。もうハチャメチャな内容だがお笑いCDとしてみればなかなかのデキといえるかも。「ロシアも強いけど大阪は負けへんどぉ」という島木譲二のシャウトには大笑いさせていただきましたよ。たはは。

お勧め度　7

終わりに

それでは今月はここまで。毎月私がチェックし損ねてしまったゲームミュージックCDのフォローを読者から募集するぞ。んでは，よろしく。

［特集］

画像創造のために

コンピュータグラフィックの発生からすでに30年が経過する。CGはその誕生のときからコンピュータとのコミュニケーション手段の一環として位置づけられていた。現在，CGによる表現力はある意味で実写を凌ぎ，すでに新たなる芸術としての一面を備えているといえよう。
自然さの追求は4次元へと拡張されている。そこには1枚絵ではなしえない表現力がある。そして技術はさらに未踏の領域へと向かいつつ，また新しいコミュニケーション手段を提示している。
コミュニケーション手段だからこそ高度なCGはもっと軽いものでなくてはならない。そのためにも造形手法は絶えず磨かれねばならない。仮想現実を実現すること，現実を再構成することはひとつの通過点にすぎない。その先に広がる世界を作ることこそがCGの目的である。

CONTENTS

5大元素別造形法講座

自然物表現の手法を探る

Nakano Shuichi
中野 修一

地形表示から始まって自然物のいろいろを順を追って再現してみることにしよう。どれも単純なプログラムだが，工夫次第で結構手軽にさまざまなものが表現できることがわかるはずだ。

新たなるツールに向けて

普通の人がコンピュータでグラフィックを扱う場合，モデリングという問題がもっとも大きな壁となります。ひとつの原因は扱いやすいモデラがなかなかみつからないということでしょう。しかし，それを考慮しても，モデリングという作業の本質はかなりの技術と地道な作業を強いるものとなっています。

2Dのペイントソフトにしても，実際に華麗にマウスを操ることができる人というのはごく限られています。この場合はツールは概ね優秀ですから，もっぱらユーザー側の責任ということになります。

従来のグラフィックツールやモデリングツールはいかにユーザーの意図するところにものを置くかという「手」の延長としての性格が強いように思われます。ユーザーの思った場所から線を引いたり，思った場所に球を置いたりといった具合です。

Z's-EXでは微分処理やフレア，ランダムフラクタルといった特殊なエフェクト関係のプログラムがありました。これらはそれぞれ，彫刻調のもの，光，雲などを手軽に表現することができます。紙やキャンバス上に描かれていた絵画をコンピュータ画面上に置き換えたという画材の延長としてのコンピュータグラフィックから，特定の処理をもってさらに積極的にグラフィック制作を支援するためのツールとしての性格を強めているといっていいでしょう。

グラフィックツールMATIERが好評なのもツール自体の操作性以上に，立体ペイントやメッシュ変形，球体マッピングなどの強力な付加機能の存在がありがたがられているからのようです。

では，このような「ユーザーが直接手を下さない」かたちのグラフィックツールはどこまで発展していくのでしょうか？

何年か前に松下電器が言葉のイメージをグラフィック化するシステムを発表したことがあります。これは「テーブルの上の2つの林檎」といった，語彙に対する図案のデータベースと意味解釈のシステムだったようですが，基本的な方向は究極のグラフィックシステムの目指すところと同一なのかもしれません。聖書でも最初に言葉があり，次に天地が創造されるのですから。

たとえば，なにかの木であるとか，家であるとかいったデータは単独でも使用できますが，組み合わせることによってより多彩な景色を構成します。重要なのはデータベースの整備でしょう。

データベースを3Dデータをもとにしたものに変え，レンダリング方法やフィルタリングを変えていけば，いずれは表現方法までも規定することができるでしょう。それはさらに複雑な形容をともなったイメージを生成できる，ということを示しています。「鬱蒼とした森の中，木もれ日を浴びて……」とかいったイメージも，やがて具体化されるようになることでしょう。

自然物とフラクタル

ひと昔前のCG業界では自然物表現が研究の大焦点として挙げられていました。その結果，現在ではワークステーションクラスのCGマシン上でさまざまな自然物表現が可能になっています。パソコンがその恩恵を受けるまでには至っていないのですが，処理時間を除けばだいたい同等のことはできるはずなのです。そこで今回は自然物表現のためのアプローチについて考えてみることにしましょう。

自然物を見て自然だと感じるのはいったいどういうところに起因するものでしょうか。たとえば地形にしても，でたらめに起伏を作ると人は地形とは認めてくれませんし，規則的すぎると人工物と判断します。

実際に自然物を見ると規則的な部分とランダムな部分が入り混じっているように思われます。それを解析してみて非整数次元空間で相似性の特徴を示す場合，それは「フラクタル」であるといわれます。調べてみると自然物の多くがフラクタルの特徴を持つ形状となっているそうです。

プログラムではっきりフラクタルの兆候を持つ図形を作成するための手法はすでに確立されており，再帰的定義を持った自己相似図形がフラクタルになることが一般に知られています。しかしそのような図形がフラクタルだからといって「自然」な形状とは限らないということは以前にも話しました。

ここでは「どんな手を使っても人間の目で見てそれっぽかったらよしとする」という当たり前の方針で進めます。最初からフラクタルの範囲内でとかやってると人間が小さくなります。よって理論的な究明や検証などは一切行いません。正確なシミュレーションなんてもってのほかです（できるにこしたことはないが）。

再帰プログラムの基礎

とはいえ，再帰図形についてなにも説明しないというのも問題があります。ここでは直接再帰図形を扱うものは少ないのですが，再帰処理の基本的な考え方から解説してみましょう。いずれ樹木の生成のところで必要になる知識ですから。

さて，再帰というのは自分自身を呼び出すことを意味しています。なんだかエラーか無限ループになりそうな処理ですね。X-BASICではそのような処理もサポートされています。

```
func recursive(p)
while p>1
    recursive(p/2)
endwhile
```

endfunc

これはプログラムの動作を遅くするための関数ですが（素直な人はなにもしない関数と表現するかもしれない），関数定義の内部で自分を呼び出しています。処理としては何回かループを回るだけで面白味はまるでありません。

確かに言語によっては再帰処理は単純にループを構成する目的で使用されたりもします。しかし，再帰を面白くするかどうかはローカル変数の活用いかんにかかっています。

再帰処理を使ったプログラムはリスト8の樹木生成です。ここでtree()という処理はどのような動作をするのか想像してみてください。

この手のプログラムは実際に実行してみると画面のでたらめな位置を描画しているようにも見えますがちゃんと処理を追っていくと辻褄が合っています。さっきまでこっちを集中的に描いていたのに，なにを頼りにほかの位置に移動していくのだろうと不思議に思う人もいるでしょう。

再帰処理は自分の内部に自分の呼び出しがありますから，なにかの処理を始めると必ず「やりかけ」のまま新しい仕事を始めます。以下同様にたらい回しにされるのですが，どこかで不意に処理を完了させるときがきます。たぶんきます。すると，今度はたらい回しにされたのと逆の順番にやり残した仕事を仕上げて帰っていきます。お利口さんなことにやりかけた仕事はすべて覚えているのです。これを可能にしているのがローカル変数です。

ローカル変数はそれを呼び出した関数が終了しないうちは有効ですから，引数として再帰呼び出しのときに使うことができます。これにより実に多彩な処理ができるのです。

ローカル変数の使えない処理系で再帰処理を行う場合はグローバル変数の配列などを使うのですが，再帰処理は階層が深くなると末端の処理数が鼠算的に膨れあがりますから，かなり大きな配列を用意しておかなければなりません。ローカル変数は必要がなくなった時点で自動的に消去されますから，常に最適な量の変数エリアしか使用されないので気軽に複雑な処理も記述できます。

で，再帰処理の記述上の注意ですが，もともと再帰処理は関数定義の途中で定義中のその関数を使ったり，関数定義の中で使われている関数の定義部分でお互いに呼び出しあったりするという，プログラムに慣れていない人が頭から1行ずつ順番に処理を追っていくと必ず混乱するような表記になっています。

混乱を避けるためには「まず終了条件を書く」というのが鉄則です。最悪の場合無限ループに陥りますからこれは必須事項でしょう。

ほとんど同じ処理なんだけど単純なループでは組めない処理のうちのいくつかは再帰を使うと実に簡単に記述できます。どのようなときに有効かは慣れてくれば自然にわかるようになるでしょう。

地の章

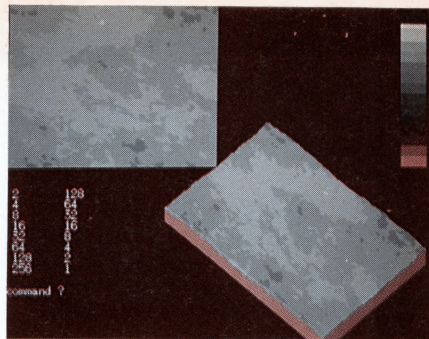
それでは本題の自然物表現のグラフィック処理に突入します。最初は「地」。要するに地形表示です。1992年12月号で発表したプログラムの発展版なのですが，以前のものを入力された方は新しく追加された関数だけを追っていけば最小限の変更だけですみます（リスト1）。

生成部分などは若干修正されています。たぶん乱数部分以外はいじらなくてもいいでしょう。アルゴリズムの都合上，矩形の境界がはっきり出てしまい困っていたのですが，システムの乱数を使うのをやめたらとたんに改善されました。乱数ルーチンはインタプリタでは動きませんので注意してください（ちょっと危ないか）。

また，それぞれの関数はほとんど独立していますので操作部のメインループさえ調整すれば必要な機能だけを残してあとは入力しなくてもかまいません。

本当は誰かほかの人にやってもらおうと思っていたのですが，都合がつきませんでした。前回はショートプログラムということもあり，必要最小限の機能しか盛り込んでいませんでした（でもないか……）。まあ，とにかくやりたいことの全貌が見えていなかったので今回は応用法についてまとめてみましょう。

前回のプログラムに必要なものはなんといっても3D表示です。といっても3D表示部分までX-BASICで記述するのも大変です。最初は疑似3D表示で斜め手前の画素との単純な差を取り，傾斜角を算定したテーブルで陰影づけをしてみようかと思っていたのですが，すでにパレットを全部使っているのでやめました。それらしく見えても疑似はあくまで疑似です。

結局，ありがちな話ですが，ここではDōGA CGAシステムでレンダリングするためのファイル出力を加えてみました。読

山を造る

者のすべてがDōGA CGAシステムのユーザーではありませんし，個人的にはDōGAシステムの使用規定範囲がいまいち明確でないなど納得のいかない点もあるのですが，適当なものがほかにありませんのでCGAシステム用のものをサンプルとします。C-TRACE用のデータにするなどの変更もおそらく困難ではありませんので，必要な方は各自で変更してください。

えっと，厳密にいえばこの記事も営利利用の一環ということになるのでしょうが，まあ，それに見合うくらいのカンパはしているので大目に見てもらいましょう。

さて，出力されるのは126×126＝15876ポリゴンの形状データです。便宜上ファイル名はTEST.SUFに固定されています。ついでにマッピング用のUV座標も加えておきましたので，レンダリングには6Mバイト程度のメモリが必要と思われます（データ量が1.6MバイトくらいになりますのでハードディスクかRAMディスクも必要です）。リスト中で注釈が付加してある行を殺し，改行を加えるとマッピングなしのデータが出力できます（uvpolyはpolyに変えること）。これならば4Mバイトのメモリとフロッピーディスクだけでもなんとかレンダリングできるかもしれません。

そんなにメモリがないという人は素直に出力するデータを小さくしてください。はっきりいって，こんなにたくさんデータが必要だとは思いません。しかし，こういうことをやるには十分なメモリが必須ですから，メモリを増設するほうをおすすめします。今回はメモリ12Mバイトのマシン上で開発しましたので，極端に大きな配列などを平気で使っています。最低4Mバイトのメモリとコンパイラが前提となっていると考えてください。

CGAシステムはほとんど使ったことがなかったので，形状データができたらもっぱらAUTOが出力するファイルをいじっていました。よくわからなくても，

AUTO /Z120 TEST.SUF

リスト1

```
10 int a,b,c,d,i,j,k,x,y,c0,siz1=64,siz2=256,com,pk,rev,seed
20 int p1(15),p2(15,2)
30 char aa(65535)
40 str dm
50 int cl1(15)={15,75,80,103,135,138,140,142
60             ,145,148,153,159,167,180,199,255}:/*雲用
70 int cl2(15,2)={125,30,20, 125,30,20, 125,30,20, 125,30,20
80               ,122,29,22, 116,29,22, 112,23,28, 106,20,30
90               ,106,15,30, 107,11,30, 110, 8,30, 110, 5,30
100              ,110, 4,31, 110, 2,31, 110, 1,31, 110, 0,31}
110 int at1(15)={15,35,42,63,79,95,111,127
120             ,143,159,175,191,207,223,239,255}:/*地形用
130 int at2(15,2)={130,30,15, 130,30,20,  20, 5,30,  37,28,25
140              , 44,21,23,  44,21,21,  55,20,19,  55,20,18
150              , 58,30,16,  67,24,12,  70,20,11,  80,20, 9
160              , 86,15, 7,  16,28,10,  16,25,15,  16,30,20}
170 screen 1,2,1,1 :console 0,31,0
180 atpal()
190 for i=0 to 255:line(480,i,511,i,i):next
200 repeat  :/*メインルーチン---------------------------------
210   cls
220   fill(0,0,siz2-1,siz2-1,150)
230   a=1
240   locate 0,18
250   if inkey$(0)<>chr$(27) then {
260     repeat                    :/*描画部メインループ
270       a=a*2:b=siz2/a
280       get(0,0,siz2-1,siz2-1,aa)
290       for i=0 to a-1
300         for j=0 to a-1
310           map()
320         next
330         if inkey$(0)="@" then break
340       next
350       print a,b
360     until b<2  :/*本当は2 デバッグ時は9
370   } else img_load("height.gm0",0,0)
380   print:print"command ?"
390   repeat                    :/*制御部メインループ
400     com=asc(inkey$(0))
410     switch com
420       case 's'  :gl3save():break
430       case 'd'  :doga_out():break
440       case 'm'  :map2():break
450       case 'l'  :p_load():break
460       case 'p'  :pers():break
470       case 't'  :smooth():break
480       case 'h'  :img_save("height.gm0",0,0):break
490       case '4'
500       case &H1D :pk=pk-2:if pk<-256 then pk=-256
510       case '6'
520       case &H1C :pk=pk+1:palset(pk):break
530       case '2'
540       case &H1F :atpal():break
550       case '8'
560       case &H1E :clpal():break
570       case 13 :rev=(rev=0):palset(pk)
580     endswitch
590     if com='q' then break
600   until com='n'
610 until com='q'
620 end :/*================================================
630 func map()         :/*地形作成ルーチン---------------------
640 int x0,y0,x1,y1,sw,cl
650   x=j*b:y=i*b
660   k=(sqr(b/3)*2)+1
670   sc=rndom(k)+20-k/2
680   for sw=0 to 3
690     x1=(rndom(b)*2+x-b+siz2)
700     y1=(rndom(b)*2+y-b+siz2)
710     cl=cl+(aa((x1 mod siz2)+(y1 mod siz2)*siz2)*sc+10)/20
720   next
730   c=cl/2
740   if c<1   then c=1
750   if c>511 then c=511
760   c=(aa(x+y*siz2)+c+1)/3
770   fill(x,y,x+b-1,y+b-1,c)
780 endfunc
790 func palset(k)         :/*パレット設定メイン関数----------
800 int p,i,j,m
810   p=1
820   for i=0 to 15
830     for j=p to p1(i)
840       m=(j+k+256)mod 256
850       if rev<>0 then m=256-m
860       if m<>0 then palet(m,hsv(p2(i,0),p2(i,1),p2(i,2)))
870     next
880     p=p1(i)+1
890   next
900   for i=0 to 63 :dm=inkey$(0):next
910 endfunc
920 func atpal()         :/*地形用パレット設定----------------
930   for i=0 to 15:p1(i)=at1(i):next
940   for i=0 to 15:for j=0 to 2:p2(i,j)=at2(i,j):next:next
950   pk=0:palset(0)
960 endfunc
970 func clpal()         :/*雲用パレット設定------------------
980   for i=0 to 15:p1(i)=cl1(i):next
990   for i=0 to 15:for j=0 to 2:p2(i,j)=cl2(i,j):next:next
1000  pk=0:palset(0)
1010 endfunc
1020 func gl3save()        :/*グラフィックセーブルーチン------
1030 int buf(256),i,j,fn,c1,pp,c2
1040   fn=fopen("maptest.gl3","c")
1050   for i=0 to 255
1060     for j=0 to 255
1070       c1=(point(j,i)+pk+256)mod 256:pp=-1
1080       if rev<>0 then c1=256-c1
1090       repeat:pp=pp+1:until p1(pp)>=c1
1100       c2=hsv(p2(pp,0),p2(pp,1),p2(pp,2))
1110       buf(j)=(c2 shl 16)+c2
1120     next
1130     fwrite(buf,256,fn):fwrite(buf,256,fn)
1140     locate 55,1:print i*2
1150   next
1160   locate 55,1:print"       "
1170   fclose(fn)
1180 endfunc
1190 func pers()          :/*エセ3D表示ルーチン-----------------
1200 int ii,jj
1210   for i=0 to siz2-2
1220     ii=i/2
1230     for j=0 to siz2*3/4-1
1240       a=point(j*4/3,i  )
1250       line(320+j-ii,210+j+ii,320+j-ii,210+j+ii-a/4,a)
1260     next
1270     line(320+j-ii,210+j+ii,320+j-ii,210+j+ii-a/4+1,230)
1280   next
1290   i=siz2-1:ii=i/2
1300   for j=0 to siz2*3/4-1
1310     a=point(j*4/3,i  )
1320     line(320+j-ii,210+j+ii,320+j-ii,210+j+ii-a/4+1,250)
1330   next
1340 endfunc
1350 /* DoGAファイル出力---------------------------
1360 func doga_out()
1370 int fn,x,y,z,c
1380 str cr,t
1390 cr=chr$(13)+chr$(10) :t=chr$(9)
1400 fn=fopen("test.suf","c")
1410 fwrites("obj suf test {"+cr,fn)
1420 fwrites("atr test"+cr,fn)
1430 for y=0 to 126
1440   for x=0 to 126
1450     fwrites("prim uvpoly ( "+cr,fn) :/* または poly
1460     z=point(x*2,y*2)-128
1470     fwrites(str$(x*4-256)+t+str$(y*4)+t+str$(z),fn)
1480     fwrites(t+str$(x*4)+t+str$(y*4)+cr,fn)  :/*
1490     z=point(x*2+2,y*2)-128
1500     fwrites(str$(x*4+4-256)+t+str$(y*4)+t+str$(z),fn)
1510     fwrites(t+str$(x*4+3)+t+str$(y*4)+cr,fn)
1520     z=point(x*2+2,y*2+2)-128                :/*
1530     fwrites(str$(x*4-252)+t+str$(y*4+4)+t+str$(z),fn)
1540     fwrites(t+str$(x*4+3)+t+str$(y*4+3)+cr,fn)
1550     z=point(x*2,y*2+2)-128                  :/*
1560     fwrites(str$(x*4-256)+t+str$(y*4+4)+t+str$(z),fn)
1570     fwrites(t+str$(x*4)+t+str$(y*4+3)+cr,fn)
1580     fwrites(")"+cr,fn)
1590   next
1600 next
1610 fwrites("}"+cr,fn)
1620 fclose(fn)
1630 endfunc
1640 /* 画像の一部を合成して張り付ける------------------------
1650 func map2()
1660 char mm(65535)
1670 int i,j,k,l,m,n,o,p,q,r
1680 get(0,0,siz1-1,siz1-1,mm)
1690 r=255:q=0
1700 for i=0 to siz1*siz1-1
1710   if mm(i)>r then r=mm(i)
1720   if mm(i)<q then q=mm(i)
1730 next
1740 r=(r+q)/2
1750 for i=0 to siz2-2
1760   m=i mod 4:p=(i¥4)*64
1770   for j=0 to siz2-1
1780     n=j mod 4
1790     k=mm(j/4+  p)+(mm(j/4+  p+64)-mm(j/4+  p))*m/3
1800     l=mm(j/4+1+p)+(mm(j/4+1+p+64)-mm(j/4+1+p))*m/3
1810     o=((k+(l-k)*n/3)-r)*3+point(j,i)
1820     if o<1 then o=1
1830     if o>255 then o=255
1840     pset(j,i,o)
1850   next
1860 next
1870 endfunc
1880 func p_load():/* 画像読み込み---------------------------
1890 int i,j,p,q,r
1900 char mm(65535)
1910 img_load("test.gm0",0,256)
1920 get(0,256,255,511,mm)
1930 for i=0 to siz2*siz2-1
1940   if mm(i)>r then r=mm(i)
1950   if mm(i)<q then q=mm(i)
1960 next
1970 put(0,256,255,511,mm)
1980 q=(q+r)/2
1990 for i=0 to siz2-1
2000   for j=0 to siz2-1
2010     r=((point(j,i+256)-q)+point(j,i)+256) mod 255
2020     pset(j,i,r+1)
2030   next
2040 next
2050 endfunc
2060 func rndom(r) :/*  乱数発生------------------------------
2070 seed=seed*17137+4287     :/*コンパイルするときのみ使用
2080 seed=(seed shl 8) xor (seed shr 8)
2090 return(seed mod r)
2100 endfunc
2110 func smooth():/*平滑化---------------------------------
2120 int i,j,k
2130 for i=1 to siz2-2
2140   for j=1 to siz2-2
2150     k=point(j,i)+point(j+1,i)+point(j-1,i)
2160     k=k+point(j,i-1)+point(j+1,i-1)+point(j-1,i-1)
2170     k=k+point(j,i+1)+point(j+1,i+1)+point(j-1,i+1)
2180     pset(j,i,k/9)
2190   next
2200 next
2210 endfunc
```

のようにするだけで全部面倒見てくれるので便利です。あとは待つだけ……。

スムースシェーディングをかけるときは，

　SHADE TEST.SUF

を事前に実行しておき，

　AUTO /Z120 /G TEST.SUF

のようにします。なお，視点の位置は原点の近く，視線は水平方向というのが推奨されております。TEST.FSCをいじってみてください。

このようにして3D表示を行ってみて感じるのは，

「平地が弱い」

ということです。山も丸すぎるきらいがあります。まあこれは好みの問題です。あとは各自の対応にまかせますが，フィルタをかけて高さの評価率を変えてやればいいでしょう。たとえば図1のような関数を各点にかけていくわけです。

このようなフィルタをいくつか用意することで一度作成したデータを何倍にも活用することができるのです。

●任意の地形生成に向けて

本来このプログラムが出力できる高さの範囲は理論上1～255までの255段階ですが，基本出力でははるかに狭い範囲のデータしか出力しません。これはもともとほかの画像データを基本地形としたユーザー指定の地形が作れるシステムにしたかったからです。

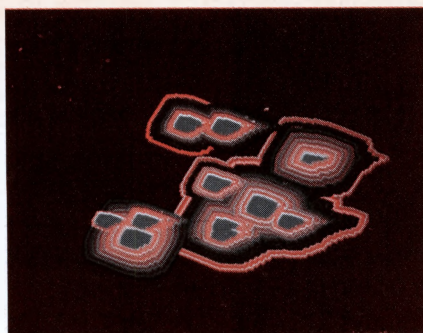
そのために加えた関数がpload()です。これは256×256ドット256色(GM0)のデータを高さのテーブルとして読み込み，基本地形と合成します。なお，便宜上ファイル名はTEST.GM0に固定してあります。

たとえばリスト2のプログラムを使えば，PIC画像を明度データに従ってこのシステムで読み込むことのできる形式に変換します。グラフィックエディタで適当にグラデーションをかけて絵を描けばそこが山になります(明るい部分が高くなる)。なお，こうして作った地形は往々にして段差が目立ちますので，画像をちょっとだけ平らにするsmooth()も用意してみました(単純平滑化)。

●再帰的合成

とはいったものの，グラフィックエディタで合成用に作成した絵が自然な感じに合成されるようになるにはかなりの修練が必要と思われます。合成法をもっと考えてやればいいのですが（たとえば一定以上の段差はできなくするとか……），これも好みの問題として放っておきます。

簡易山エディタ

ワイヤーフレームのほうがいい感じ？

前述のように単純に生成された地形はのっぺり気味ですので，不自然でない輝度変化を加える（その1）としてこのプログラム自体で生成されたパターンを使用することにします。

関数map2()は画面の左隅64×64ドットの領域を引き伸ばして全体に張り付けるためのものです。拡大した途中のポイントは線形に補間していますのでガタガタにはなりません。合成時の倍率は好みで変更してください。ちなみにカラーページの例は倍率4です（デフォルトは3）。

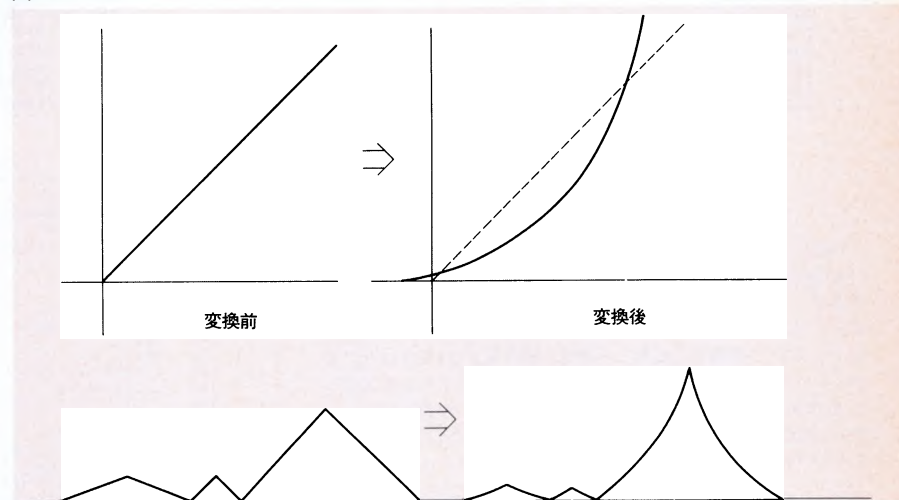
さて，以前，これは「自己相似でないアルゴリズムの再帰図形」だと説明したと思うのですが，できあがった画像に手軽にメリハリをつけるため地形の一部を拡大して合成する手法を試してみました。これを進めればまっとうな再帰図形になります。

●もうちょっとなんとか……

もう少しちゃんと指定の位置に山を作りたいという要望（誰のだ？）のために簡易山エディタを作成しました（リスト3）。

「頂点の隣から右手法でその周りを回り，1周したら1ドット離れて処理を繰り返す」というものですが，途中に乱数でゴミを加えていきますのである程度変化に富んだ形になるはずです。アルゴリズムは明快ですから(操作次第ですが)，滑らかな輝度変化になります。

エディタのユーザーインタフェイスは極度に簡略化されています。まずマウスで位置を指定し，適当な高さを数値で入力して

図1

リスト2

```
 10 /* PIC画像を高さデータに変換する
 20 int i,j,k,l,m,n
 30 char a(65535)
 40 screen 1,3,1,1
 50 pic_load("test.pic",0,0)
 60 for i=0 to 255
 70   for j=0 to 255
 80     m=point(j*2,i*2)
 90     n=m/2048*12+(m mod 2048)/64*8+((m mod 64)¥2)*4
100     pset(j,i,rgb(n/24,n/24,n/24)):a(i*256+j)=n/6+64
110   next
120 next
130 i=fopen("test.gm0","c")
140 fwrite(a,65536,i)
150 fclose(i)
```

いきます。数値0で終了，セーブされます。このデータもGM0でセーブされます。

天の章

雲については基本的に地の章と同じです。使用するツールも同じでパレットを切り替えるだけです。読み込む画像を工夫すれば，きっとイワシ雲のようなものも作成できるでしょう。よって省略。

星空は……省略。

そのほか空気的なものを表現するといっても，たいてい透明ですからあまり考えることはありません。存在を強調するには空気遠近法をかけるだけです。

ここでは個人的に以前から作りたかった陽炎処理について考えてみましょう。

陽炎は夏の暑い日に遠くがゆらゆらとかすむあの現象です。

熱源からの温度差によって発生する乱流とその密度差による空気の屈折率の変化，そしてそこを通過する場合の光線の軌跡から画像を正確に計算する……というのはスーパーコンピュータを使ってもできそうにない問題だというのはわかりますね。

実際使うにはそれっぽい効果というところで十分です。屈折というのがクセモノですから，レイトレーシングなら透明体の板へのバンプマッピングをアニメーションさせてやれば比較的簡単に実現できるのでしょう。場合によっては屈折率マッピング一発で終わりです。

しかし，私が行いたいのは2次元画像に対するフィルタリングです。基本的には局所的な拡大縮小処理が行われていると考えればいいのですが，自然な感じに制御する方法がいまひとつ浮かびません。ここでは屈折処理を簡易化することで簡易版の陽炎フィルタを作ってみました。

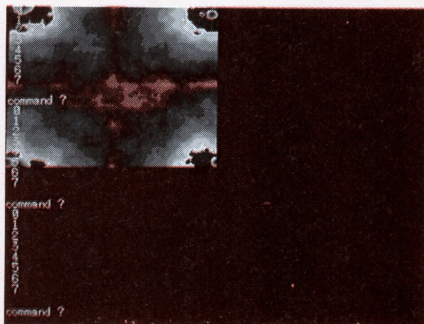
まず，空気の疎密状態を示すものとしてランダムフラクタルの雲を用意します。ある点が変換の結果どこに対応するかというのをX，Y座標の変移としてフィルタ表面の状態から算出します。例によって計算部分に確たる根拠はありません。

変換部分ができればあとはフィルタを変形するなりの処理をしながらアニメーションを行うだけです。

リスト3

```
 10 screen 0,2,1,1
 20 int i,j,k,l,m,n,d,x,y,z,a,b,h,bl,br,dm
 30 mouse(1)
 40 repeat
 50   x=rnd()*300+106:y=rnd()*300+106
 60   msstat(dm,dm,bl,br)
 70   if bl=-1 then {
 80     input z
 90     mspos(x,y)
100     sima(x,y,z)
110     }
120 until br=-1
130 img_save("test.gm0")
140 end
150 func sima(x,y,z)
160 pset(k-1,l,z+1)
170 a=x:b=y:d=0
180 for i=0 to z
190   gururi(a,b,z-i)
200 next
210 endfunc
220 func gururi(x0,y0,h)
230 a=x0:b=y0
240 repeat
250 if rnd()<0.5# then set(h)
260   move(h)
270   if rig(d)<=h then d=(d+1) mod 4:continue
280 until ((a=x0) and (b=y0))
290 a=a+1
300 endfunc
310 func rig(s)
320 int r
330 if s=0 then r=point(a-1,b)
340 if s=1 then r=point(a,b-1)
350 if s=2 then r=point(a+1,b)
360 if s=3 then r=point(a,b+1)
370 return(r)
380 endfunc
390 func move(h)
400 if rig((d+3) mod 4)<=h then {
410   pset(a,b,h)
420   if d=0 then b=b+1
430   if d=1 then a=a-1
440   if d=2 then b=b-1
450   if d=3 then a=a+1
460  } else d=(d+3) mod 4
470 endfunc
480 func set(h)
490 if d=0 then pset(a+1,b,h)
500 if d=1 then pset(a,b+1,h)
510 if d=2 then pset(a-1,b,h)
520 if d=3 then pset(a,b-1,h)
530 endfunc
```

Chris Gray氏のアルゴリズム

1992年12月号で紹介しそこねたAMIGAのFRSでよく使用されているChris Gray氏の地形生成アルゴリズムを紹介しておきます。C言語のソースからエッセンスだけ抜き出してX-BASICで記述してみました。いま見ると私の方法と基本的な構造は大差ないような気もします。「再帰的に乱数と平滑化を繰り返す」作業をいかに効率よくしかも高品質に行うかというところでそれぞれのアルゴリズムの特性が出てくるようです。定数などは，いかにも山という感じの地形を作るように設定されているようで，出力されるものについてはかなり癖も強いようです。参考までに。

```
 10 screen 1,2,1,1
 20 int j,c,i,step,nextstep,j1,j2,c1,c2
 30 char a(512,512)
 40 char range(8)={32,32,32,22,14,8,4,2}
 50 a(0,0)=0
 60 step=256
 70 for i=0 to 7
 80 print i
 90   nextstep=step/2
100   j=0
110   while j<256
120     j1=j+nextstep
130     j2=j+step
140     if j2>255 then j2=0
150     c=0
160     while c<256
170       c1=c+nextstep
180       c2=c+step
190       if c2>255 then c2=0
200       set(j,c1,i,(a(j,c)+a(j,c2)+1)/2)
210       set(j1,c,i,(a(j,c)+a(j2,c)+1)/2)
220       set(j1,c1,i,(a(j,c)+a(j,c2)+a(j2,c)+a(j2,c2)+2)/4)
230       c=c+step
240     endwhile
250     j=j+step
260   endwhile
270   step=nextstep
280 next
290 for i=0 to 255
300   for j=0 to 255
310     pset(j,i,a(j,i))
320   next
330 next
340 end
350 func set(j,c,size,height)
360 int r
370 r=range(size)
380 height=height+rnd()*r-(r+1)/2
390 a(j,c)=height
400 endfunc
```

なお，これはアニメーションデータを前提にした処理ですので一枚絵にかけても絵を破壊する以外の効果はもたらしません。特に今回は平滑化もしていませんし，かなり乱暴な変換をしていますので，あしからず。

木の章

自然物のなかでも，わりとはっきりした規則性をも備えているのが植物の構造です。これにはフラクタルと乱数を組み合わせるのがもっとも適していると思われます。

木は３Ｄで描画すると応用範囲が非常に広がりますので，ここでもCGAシステムを使用することにします。CGAシステムのマニュアルをぱらぱらと見ているとフレームソースファイルを使うのがもっとも有効そうです。

フレームソースというのは，アニメーション作成の基本的な物体配置などを記述するものです。フレームソースは一種の言語です。たいていの処理ならこの内部で記述できると思われるくらいの多彩な機能を持っているようです。FFというツールを使えばこれをもとに実際の１カットずつの画面に適合したフレームファイルを生成してくれますので，それをRENDにかけていくわけです。

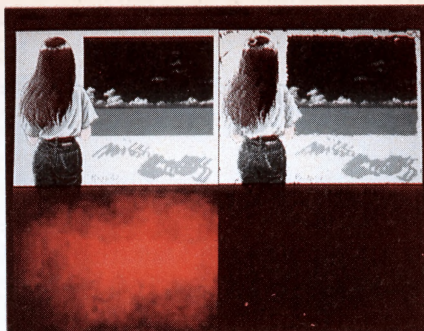
ここで作成するプログラムはX-BASICのファイル処理で記述し，フレームソースを出力します。DōGAの達人になるとこれくらいの処理はフレームソースだけで記述できるのかもしれませんが，再帰処理ができるかどうかマニュアルに明記されてなかったことや乱数ルーチンを組むのが嫌だったこと，などの理由からX-BASICのプログラムとして作成しました。

描画に使用する基本アイテムを枝と葉だけに限定して，これらの組み合わせで表現することにします。最低限これらの形状ファイルはあらかじめ用意しておかねばなりません。私はCADに慣れていないので直接形状ファイルを書きました。とりあえずリスト５～７のファイルはエディタから打ち込んでおいてください。この関数群はフレームソースを出力するためのものですから，実際にできた図形を眺めるためには，

```
FF tree.fsc
REND tree.flm tree.atr leaf.suf
stem.suf
```

のようにしてレンダリングし，HANIMで再生します。詳しくはDōGA CGAシステムのマニュアルをご覧ください。

陽炎処理

変換後の画像

さて，この樹木作成プログラムはこれ自体で一種の言語のようなものを構成しています。関数によってBASICを樹木の生成向きに拡張した感じに仕上げました。

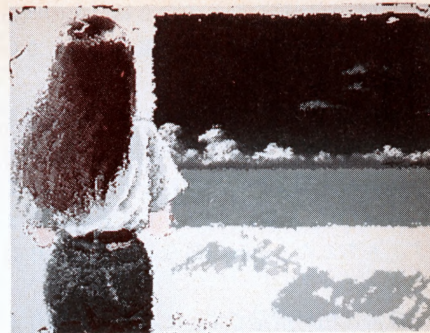
生成規則はtree()の内部に記述します。基本的な考え方を解説しておきましょう。

stem()とleaf()はそれぞれ茎と葉の部品を置くという指定です。位置と取り付け角度はあらかじめ設定されているものとします。

●角度指定

まずは取り付け角度からです。角度の指定はＺ軸周りの回転とＸ軸周りの回転だけで指定します。Ｚ軸周りの回転は茎の周りのどの位置かを指定し，Ｘ軸周りの回転は茎自体の角度を設定します。必ずＺ軸から回し，回転させる部分が終わったら，手動で元の角度に戻していくようにしてください。

たとえば双子葉植物の芽のように枝の先から２枚の葉が120度の角度で出ているという場合は，

```
stem()
move(10)
/*rotz(0)
    rotx(120)
        leaf()
    rotx(-120)
rotz(180)
    rotx(120)
        leaf()
    rotx(-120)
rotz(-180)
```

のように記述します。

この例の内容はまず，茎を置き，茎の先端位置までポインタを移動させ，その位置から120度の角度で葉をつけ，その反対側にも葉をつけ加えるというものです。三つ葉にするにはrotzを120度ずつの回転にして葉の処理をもうひとつ加えていけばいいわけです。

もともとがBASICですから，

```
for i=1 to n
    rotx(120)
        leaf()
    rotx(-120)
    rotz(360/n)
next
```

のような記述でｎ葉のものができあがります。ここではＺ軸の回転で角度を戻すことを省略しましたが，結果的に処理の前後で360度回転していることになるので辻褄が合うのです。このようにＺ軸周りの回転の場合は角度の戻しが省略できる場合もあり

リスト４

```
 10 /* mirage.bas
 20 /*     陽炎処理をアニメーションにする
 30 /*
 40 int i,j,x,y,c,k,m
 50 input k     :/*30～50くらいか?
 60 screen 1,3,1,1
 70 console 0,31,0
 80 pic_load("test.pic",0,0)
 90 pic_load("mirage_filter.pic",0,256)
100 for m=0 to 30
110 for j=1 to 254
120   for i=1 to 254
130     y=(point(i+m,j+255) mod 2048)-(point(i+m,j+257)mod 2048)
140     y=y+((point(i+m-1,j+255)mod 2048)+(point(i+m+1,j+255)mod 2048))*0.7#
150     y=y-((point(i+m-1,j+257)mod 2048)+(point(i+m+1,j+257)mod 2048))*0.7#
160     x=(point(i-1+m,j+256)mod 2048)-(point(i+m+1,j+256)mod 2048)
170     x=x+((point(i+m-1,j+255)mod 2048)-(point(i+m+1,j+255)mod 2048))*0.7#
180     x=x+((point(i+m-1,j+257)mod 2048)-(point(i+m+1,j+257)mod 2048))*0.7#
190     c=point(i+x/k,j+y/k)
200     pset(i+256,j,c)
210   next
220 next
230 pic_save("m"+right$("00"+str$(m),3)+".pic",256,0,511,255)
240 next
```

ますが，rotx()に関しては２軸以上の回転となるので省略はなるべく避けてください。

●move()について

move()は次の部品を取り付ける位置を指定するものと思ってください。形状データが茎の先端を10の位置にとっているので，move(10)で先端から伸ばすことができます。もちろん，move(5)なら茎の真ん中になります。

位置は相対指定です。このあとの操作がすべてこの影響を受けますので，真ん中から分岐してさらに先にも続けるといった場合はmove(5)を使わずに２つの物体を組み合わせてください。

●大きさを変える

scal()は「これから扱う物体の大きさを変えますよ」という指定です。scal()ではX，Y，Zとも同率で変化させています。

scal()はその後の操作のすべてに影響を与えますので注意して使ってください。たとえば，

　scal(0.5)
　scal(0.5)

ならば，ここ以降で扱う物体は0.25倍の大きさで作成されます。

倍率以外に，たとえば茎の長さを変えたいという場合は変数zsを直接書き換えていってください。この場合の数値は絶対指定となっています。

　zs＝zs＊1.2

のようにすれば，茎の基本形状をだんだん長細くしていくこともできます。この場合でも茎の先端に移動するときには，

　move(10)

でかまいません。

定数的な変数(？)の書き換えですから，いったん変更したらそのあとはすべてその比率で計算されます。一時的な変更ならすぐに戻すようにしましょう。

●全体の比率について

このプログラムで使用する基本形状は図のようにかなりずんぐりしたものになって

リスト5　LEAF.SUF

```
 1: obj suf leaf {
 2: atr leaf
 3: prim poly (
 4:         0        0        11
 5:         0        0        -1
 6:         4        1        3
 7:         )
 8: prim poly (
 9:         0        0        11
10:         0        0        -1
11:         -4       1        3
12:         )
13: }
```

リスト6　STEM.SUF

```
 1: obj suf stem {
 2: atr stem
 3: prim poly (
 4:         5        2        -1
 5:         5        2        10
 6:         2        5        10
 7:         2        5        -1
 8:         )
 9: prim poly (
10:         2        5        -1
11:         2        5        10
12:         -2       5        10
13:         -2       5        -1
14:         )
15: prim poly (
16:         -2       5        -1
17:         -2       5        10
18:         -5       2        10
19:         -5       2        -1
20:         )
21: prim poly (
22:         -5       2        -1
23:         -5       2        10
24:         -5       -2       10
25:         -5       -2       -1
26:         )
27: prim poly (
28:         -5       -2       -1
29:         -5       -2       10
30:         -2       -5       10
31:         -2       -5       -1
32:         )
33: prim poly (
34:         -2       -5       -1
35:         -2       -5       10
36:         2        -5       10
37:         2        -5       -1
38:         )
39: prim poly (
40:         2        -5       -1
41:         2        -5       10
42:         5        -2       10
43:         5        -2       -1
44:         )
45: prim poly (
46:         5        -2       -1
47:         5        -2       10
48:         5        2        10
49:         5        2        -1
50:         )
51: }
```

図2　形状ファイルの実際

図3　植物の枝の構成例（参考）

リスト7　TREE.ATR

```
 1: /*
 2:        Attribute Editor ver3.0
 3: */
 4:
 5: atr  leaf       {
 6:                          col ( rgb ( 0.40 0.90 0.20 ) )
 7:                          amb ( 0.25 )
 8:                          dif ( 0.30 )
 9:                          spc ( 0.40 0.55 0.50 )
10:                 }
11: atr  stem       {
12:                          col ( rgb ( 0.70 0.20 0.10 ) )
13:                          amb ( 0.25 )
14:                          dif ( 0.30 )
15:                          spc ( 0.40 0.55 0.50 )
16:                 }
```

います。

これは計算に都合のよい数値にしているだけの話で，プログラム先頭の数値を書き換えることで好みの比率で設定することができます。葉については形の変形は考慮していませんが，茎の部分についてはxs，ys，zsのそれぞれで変更します。基本的に丸い茎ならxs，ysは同じ数値を指定してください。zsは茎の長さを示しています。これらの数値は基本形状に対する倍率ですので，もともとの図形が，

半径５

長さ11（原点からは10）

で設定されているものだということを覚えておくといいでしょう。

なお，茎に対する葉の大きさはlsで変更します。

植物の構造

基本的な記述方法がわかったところで，具体的にどのようにしたら植物らしくなるのかを検討してみましょう。枝葉の発生の規則には大きく分けて２つの種類があります。１箇所から複数発生するタイプと互い違いに発生するタイプです。あとは枝の分岐発生までの長さや分岐の角度でだいたいの構造が決定します。分岐発生の有無や分岐の角度には乱数を使ったほうがいいでしょう。

厳密にいえば分岐の角度は重力や太陽の方向などを考慮に入れるべきかもしれませんが，かなりランダムに行ってもそれほどひどい結果にはなりませんのでほとんど無視してもかまいません。

基本的な記述方法がわかったら，あとは試行錯誤でより自然な形態を模索していくだけです。曲がりくねった枝なども簡単な応用で作成できるでしょう。

もちろん，これらの関数は樹木生成の専用パッケージではありませんから，空中をペアノ曲線で埋めてみたり，再帰ピラミッドを作ってみたりといった処理がきわめて簡単に記述できます（形状の確認には時間がかかりますが)。初めっから３次元タートルグラフィック関数だと説明したほうが適切だったかもしれません（もう少し関数を補足したほうがいいけど)。

火の章

火というのはCGで表現するのにいちばんやっかいなものかもしれません。燃え上がる炎の本質は乱流です。乱流といえば，

リスト８　樹木の生成

```
 10 str cr,tb
 20 cr=chr$(13)+chr$(10):tb=chr$(9)
 30 int n,f
 40 float xs=2
 50 float ys=2
 60 float zs=7
 70 float ls=11
 80 /*                       ------形式上のメインプログラム
 90      okimari()
100      tree(7)
110      owari()
120      end
130 /*
140 func tree(n)          :/*-------------ここが本体↓
150   float a,b
160   st_in()
170     if n<>0 then {
180       stem()
190       move(10)
200       scal(0.8#)
210       a=rnd()*60
220       rotz(a)
230           b=rnd()*35
240           rotx(b)
250               tree(n-1)
260           rotx(-b)
270       rotz(180)
280           b=rnd()*35
290           rotx(b)
300               tree(n-1)
310           rotx(-b)
320       rotz(-180-a)
330     } else {
340     /*rotz(0)
350           rotx(60)
360               leaf()
370           rotx(-60)
380       rotz(180)
390           rotx(60)
400               leaf()
410           rotx(-60)
420       rotz(-180)
430     }
440   st_out()
450 endfunc    :/*================================以下サブルーチン
460 func okimari()     :/*-----ファイル作成開始
470   f=fopen("tree.fsc","c")
480   fwrites("#frame ( fno , 1 , 20 )"+cr,f)
490   fwrites("fram {"+cr,f)
500   fwrites(tb+"light pal( rgb ( 1 1 1 ) 5 2 -6 )"+cr,f)
510   fwrites(tb+"{ mov (-730 0 50 ) eye deg ( 65 ) }"+cr,f)
520   fwrites(tb+"{"+cr,f)
530   fwrites(tb+"rotz( ¥fno*18¥ )"+cr,f)
540 endfunc
550 func owari()       :/*-----ファイル作成完了
560   fwrites(tb+"}"+cr,f)
570   fwrites("}"+cr,f)
580   fwrites("#endframe"+cr,f)
590   fclose(f)
600 endfunc
610 func rotx(d)       :/*-----X軸周りの回転
620   fwrites(tb+tb+"rotx("+str$(d)+")"+cr,f)
630 endfunc
640 func rotz(d)       :/*-----Z軸周りの回転
650   fwrites(tb+tb+"rotz("+str$(d)+")"+cr,f)
660 endfunc
670 func stem()        :/*-----茎の作成
680   fwrites(tb+tb+"scal ( ",f)
690   fwrites(str$(xs)+" "+str$(ys)+" "+str$(zs)+")"+cr,f)
700   fwrites(tb+tb+"obj stem"+cr,f)
710 endfunc
720 func leaf()    :/*---------葉の作成
730   fwrites(tb+tb+"scal ( ",f)
740   fwrites(str$(ls)+tb+str$(ls)+tb,f)
750   fwrites(str$(ls)+" )"+cr,f)
760   fwrites(tb+tb+tb+"obj leaf"+cr,f)
770   fwrites(tb+tb+"scal ( ",f)
780   fwrites(str$(1#/ls)+tb+str$(1#/ls)+tb,f)
790   fwrites(str$(1#/ls)+" )"+cr,f)
800 endfunc
810 func move(z)  :/*---------原点移動
820   fwrites(tb+tb+"mov ( 0 0 "+str$(z)+" )"+cr,f)
830   fwrites(tb+tb+"scal ( ",f)
840   fwrites(str$(1/xs)+tb+str$(1/ys)+tb+str$(1/zs)+")"+cr,f)
850 endfunc
860 func scal(s;float):/*-----全体の拡大/縮小
870   fwrites(tb+tb+"scal ( ",f)
880   fwrites(str$(s)+tb+str$(s)+tb,f)
890   fwrites(str$(s)+" )"+cr,f)
900 endfunc
910 func st_in()  :/*---------構造体作成
920   fwrites(tb+tb+"{ "+cr,f)
930 endfunc
940 func st_out() :/*---------構造体終了
950   fwrites(tb+tb+"} "+cr,f)
960 endfunc
```

単純な分岐による樹

ちょっと複雑にしてみた

最近流行のカオスです。この手のものは流体力学で追っていくよりはカオスとして扱うことのほうが適しているらしい……のですが，カオス関係の書籍も見て（読んでではなく）みましたが，応用する手法というのが思いつきません。まあ文系の人間に数式のいっぱい入った本を理解しろというほうが無理なのでしょうが。

概ねのところ，カオスというのは複雑な事象を分析する際に厳密な計算をあきらめて乱数と確率を持ち込むための口実なのかもしれません。よくわかりません。ごめんなさい。

で，わからないものはさっさと捨てて，適当な理屈をこじつけてみます。

煙草の煙を吐き出した場合，流れとそれ以外の部分との衝突(?)からいくつもの渦が発生して分散していきます。全体としては普通の空気と煙の部分との入り混じった流れを持つ煙の塊になります。各部が丸くなっているのは乱流，つまり渦が発生していることを示しています。

渦がどうしてできるのかというと，流れと流れの隙間で物理的な力が働いているのかもしれませんし，複雑にからみあった空気の流れのうち，平衡した部分がコリオリの力で回り出すのかもしれません。ひょっとしたら誰か恐ろしく複雑な方程式で右辺が左辺より大きくなったとき云々と解析している問題かもしれませんが，こういったものを計算する気はさらさらありません。流体力学をパソコンでやってしまってはスーパーコンピュータに義理が立ちません。

流れから渦を求めるのは無理ですので，初めっから渦（の中心）を用意してその周りに流れを適当に作ってみます。流れというのがまたつかみがたい概念なのですが，それらしく見えるということで，空間の変形（座標の変換）で代用します。

変形する元になるものとして，ある程度それらしい図形を用意します。これにはグラデーションの球体（MATIERで作成）にランダムフラクタルをかけた「もやもや」を使用しました。球は円形のグラデーションでもかまいません。ランダムフラクタルはZ's-EXのものを使ってください（おや，こんなところにMATIER-EX（仮称）が……）。この「もやもや」を基本データとして処理することで炎や煙を表現してみようというのがリスト9です。

まだ実験的なものですのでいまひとつという感触ですが，渦が同じ方向であること，強さが同じであること，まだ加減がわかっていないことなどから将来的な画質の改善は期待できます。大局的にも局所的にも影響はもっと小さいほうがいいみたいです。ものが炎ですから重力の影響などのフィルタも入れないと自然なかたちにならないはずでもあります。ついでにプログラムは浮動小数点の山ですので非常に遅いです。

単なる火の1枚絵ならもっと簡単に作れるのでしょうが，最終的にほしいのはリアルなアニメーションデータです。

この方法から炎や煙のアニメーションデータを得るためには時間とともに渦の強度と位置を移動しながらデータを変換していきます。シミュレートされた結果に基づいていないアルゴリズムなのですが，かなり自由にパラメータの変更がきくはずなので「それらしいもの」くらいは生成できそうです。どの程度の数の制御点を使用すればいいのか，そもそもこれでそれらしく見えなかったらどうするのかという疑問も残りますが，将来的な課題ということにしておきましょう。

さらに，ここで用いる関数は渦巻きでなくても空間を連続関数で変換するものならなんでも試してみる価値があります。単純な渦だと無難は無難ですが特徴がはっきりしすぎるので大量の制御点が必要となるでしょう。分野はまったく違いますが，たとえていうと，サイン波合成で作る「リアルな音」というのは大変大掛かりなものになりますが，FM変調をかけると4つのサイン波だけでもかなり多彩な音が作れますね。これを考えると，空間自体になんらかのモジュレーションをかけるつもりで変な関数を用意したほうが効率がよいとも推定できます（理論的な保証はありません）。

この渦巻きはもともと，昔作ったFANT.X（Z's-EX用のフィルタ）の新しい軌跡用に考えていた方法です。単体でもそれなりの特殊フィルタとして使えると思っていたのですが，編集室で普通のグラフィック

CGAシステムを使ってみて

初めてDōGA CGAシステムに触ったのは昔アスキーで記事が連載されていた頃だったでしょうか。TAKERUでソフトを買ってきた私はCADをちょっと触ってみて，これには手を出すまいと心に決めたことを思いだします。はっきりいって使い方が全然わかりませんでした（マニュアルなしで売るってのもひどい）。

最近のCADシステムでは全体の環境はかなり充実してきましたし，作成される作品のレベルもずいぶん向上しています。しかし，新しいシステムを見てもやはりCADがほとんどかわっていないのには驚いてしまいました。「どうやったらこれであんなものが作れるんだ？」というのが正直な感想です。

市販のモデラでよいものがあればいいのですが，せいぜいZ'STRIPHONYどまりです。画面で3次元の物体をエディットするということ自体に解決しがたい問題があるのだと半ば信じ込んでいたようです。AMIGAのCaligari2を見たときはぶっ飛びました。目新しい機能というのはひとつしかないのですが，切るわ伸ばすわ回すわで3Dオブジェクトが粘土のように扱えるものだと知り，世界観が変わった気がします。

CADの機能自体は決して低くありません。しかし，モデリングという作業を3次元座標での点を結んでいくこととしてしか扱えていないことが扱いにくさの原因なのでしょう。

それでもモデラ以外の部分の完成度やシステムの柔軟性はたいしたものです。加えてRENDやFFなどの，おそらくすべてのユーザーに常用されているだろうツールについてはかなりの信頼性が確保されていると考えていいようです。考えようによってはかなり無理のあるデータを与えたにもかかわらず確実に動作してくれました。半面，WIREVIEWやKAMAではまだ脆さが目立ちます。

ちょっと凝った樹木などを作ろうとすると，フレームソースは100〜200Kバイトくらいで収まるのですがそこから生成されるフレームファイルがすぐにメガバイト単位になるので実行がかなり苦しくなります。こういった処理は本来レンダラと密結合していないと効率が悪いのです。すべてをいったん形状ファイルにし共通仕様という形で管理することは思想的に明快です。それを受け取る側がいわばRENDというブラックボックスになっている，というか現在はRENDに頼りすぎている部分が目立つのも事実です。CGAシステムは今後も進化を続けていくでしょうが，どのような回答を出すのかちょっと楽しみでもあります。

をグネグネ回してテストしていたら気持ち悪いといわれてしまいました。ちなみに回転行列のsinとかcosの符号をひとつだけ変えたり，cosをsinにしたりするともっと気持ち悪くなります。計算量の割に不確定性が増すのでうまくすれば使えるかな……。

水の章

これだけいろいろやっていると，やはり水の表現についても考えなければならないかなと……都合によりサンプルプログラムはありません。

水面を考えます。問題になるのは基本的に「波」です。サイン波合成でパターンを作り出しバンプマッピングを行う……というのが基本的な方針となります。

レイトレの作品などではうねうねと波を加えているものも多々あるのですが，実際の海はもっと複雑です。自然に近い波というのはなかなか合成しにくいものです。これは風の影響，つまり空気が水面にぶつかったり摩擦で影響を与え……つまるところ乱流が出てくるのでカオスの世界に突入してしまいます。

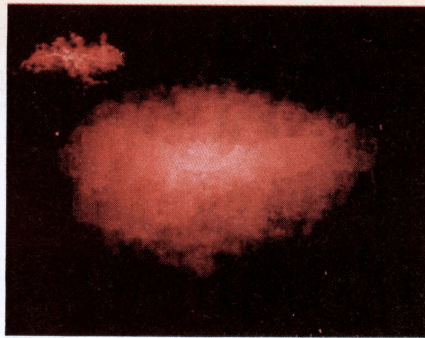
ていねいにサイン波を合成していってもあまり報われない気がして手を出さなかったのですが，少しは試してみるべきだったのでしょうか……。

いろいろな地形表示プログラムを見るとどうもできあいの絵を水面部分に張り付けることで表現しているようです。何パターンか用意しておいて張り付けるものを選べれば，それだけでたいていの問題は片付いてしまいます。

日の光によるきらめきとかの表現も疑似的なもので間に合ってしまいそうです。先ほど陽炎で使用したフィルタはそのまま使えるかもしれません。さらに，わずかなうねりをLFO的にかけてやれば文句をいう人は少ないでしょう。

これは指定された平面をいかに水面に見えるように塗り潰すかが問題となっています。いわゆるカラーマッピングとバンプマッピングです。しかし，実際にはバンプマッピングではすまない水面も存在します。

極端にいえば富嶽三十六景の「あれ」とか，打ち寄せる波関係については水面の形状自体が大きく変化してしまいます。いわゆる海のソリトンというやつです。

ソリトン（孤立波）とは広がったり拡散したりしないで粒子的に振る舞う波を意味します。強いエネルギーを持った波が特定の条件でソリトンに変化します。物理的にいろいろな特性を示すようですが，深くは追求しません。おそらく一定以上のエネルギーになると重力と水面や空気の抵抗，表面張力などのバランスが崩壊して白い波頭が発生するのでしょう。例によってパソコンで真面目に計算する馬鹿はいません（いたらご連絡ください）。

もし作れといわれたら，波頭形状の生成プログラムを適当にくっつけて波の形状を一定速度で動かすだけですませてしまうでしょう。通常の水面パターンはそのまま合成します。

最後に川。水面に発生したパターンは多少崩されながらも相対的には形を保ちつつ移動していきます。水面自体の移動とはほぼ独立に水面もアニメーションさせます。川と同じ速度でゆっくりうねっているだけでいいでしょう。ところどころに渦があって渦の位置はしっかり固定されている……という変換を行えば誰がなんといおうと川に見えるはずです。

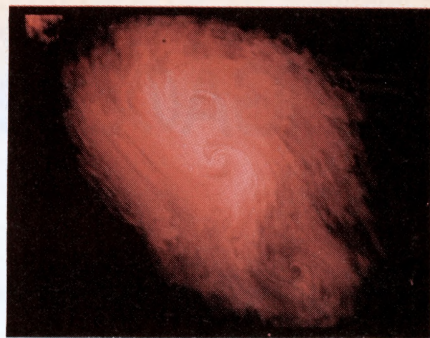
変換前の像

変換後．煙に見えるか？

最後に

私たちが感じる「自然さ」というものは（おそらく）規則性とランダム性の微妙な関係にあります。

これは自然物だけではありません。木や家の基本データをいくつか用意しておき，パラメータをちょいと変えるだけでわさわさとさまざまな道路や住宅地を生成するといったことも可能なのではないかという意見も出ています。

モデリングから解放される日はいつ訪れるのでしょうか……。

＊　＊　＊

このようにさまざまな自然物を作ってみたわけですが，アルゴリズムをひねくり出すには私流のやり方があります。まず，表現したいものについてのイメージを持つことです。なにか浮かんだら図を描きます。数式をこねくるのが苦手でして，図にして直感的に確認しないと正しいかどうかわからないからです。そしてテストプログラムをいくつか作っていきます。面白い部分から作り始め，退屈な部分は後回しです。それでもなんとかできあがり。

木の生成がありきたりになってしまったのが残念です。あ，ランダムドットで出すという手もあったんでした……。今後の課題もまだまだ多いようです。

リスト9　渦による画像の変形

```
 10 float a,b,x,y
 20 int vx(511,511)
 30 int vy(511,511)
 40 int aa(511,511)
 50 for j=0 to 511
 60   for i=0 to 511
 70     aa(i,j)=point(i,j)
 80   next
 90 next
100 for i=0 to 2
110   voltex(rand()/64,rand()/64)
120 next
130 disp()
140 end
150 func voltex(cx,cy)
160 int i,j,x,y
170 float a,b
180 y=128
190 for j=0 to 511
200 print j
210   for i=0 to 511
220     a=sqr((i-cx)*(i-cx)+(j-cy)*(j-cy))
230     b=3*3.1416#/(a/5#+0.5#)
240     k=(i-cx)*(1-1#/(a*a)):l=(j-cy)*(1-1#/(a*a))
250     x=k*cos(b)-l*sin(b)
260     y=l*cos(b)+k*sin(b)
270     vx(i,j)=vx(i,j)+x+cx-i
280     vy(i,j)=vx(i,j)+y+cy-j
290   next
300 next
310 endfunc
320 func disp()
330 for j=0 to 511
340   for i=0 to 511
350     x=vx(i,j)+i
360     y=vy(i,j)+j
370     if x<0 then x=0
380     if x>511 then x=511
390     if y<0 then y=0
400     if y>511 then y=511
410     pset(i,j,aa(x,y))
420   next
430 next
440 endfunc
```

パソコンでは数少ないフラクタル地形作成ツール

AMIGAのScenery Animator&VISTA PRO

Akikawa Ryou
秋川　涼

山や雲などの自然画像を計算で描く。フラクタルと呼ばれる理論は最近になって一般的なものになってきたが，応用した地形作成ツールはパソコンレベルではまだ少ない。実用に耐えうるのは，現在この2本ぐらいであろう。

ある一定の法則のもとに数値が弾き出され，それが我々の目に見えるものとなって現れる。たとえば，マンデルブロやジュリア集合。高次元方程式のグラフなどもそうだ。マンデルブロなどは特にコンピュータ関係者の興味をひくものらしく，フリーウェアなどで頻繁に見かける。が，いかんせん実用性に乏しい。

皆さんはフラクタル幾何学という言葉を聞いたことがあるだろうか。形状を全体的に捉えて作成するのではなく，ある一定の規則（計算式）に沿って生成していくという理論である。

この規則にはさまざまなものが考えられているが，身近で理解できそうなところでは自己相似性といわれる特性が挙げられる。木の一部を取り出すと，小さいながらも木全体の形状と似通っているときがある。自然界の事物が必ずしもそういう法則に従っているわけでないが，これを利用すると木の全体図を与えるだけで，細かい部分まで自動的に計算して作らせることができる。山もまたしかりである。

今回紹介するソフトはともにAMIGA用のソフトで，「Scenery Animator」と「VISTA PRO」という地形作成ツールである。フラクタル理論を応用して地形を描かせるという目的は共通だが，内容的には異なったものとなっている。ワークステーション向けにも似たような製品はあるが，もちろん価格が尋常ではない。MacintoshやPC-9801にもあるようだが，価格と機能がイマイチらしい。

というわけで，地形作成ソフトではこの２本のソフトを押さえておけばいいだろうという結論に達した。では，さっそく１本ずつ紹介していこう。

Scenery Animator(Natural Graphics)

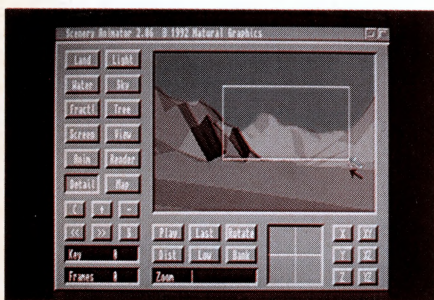
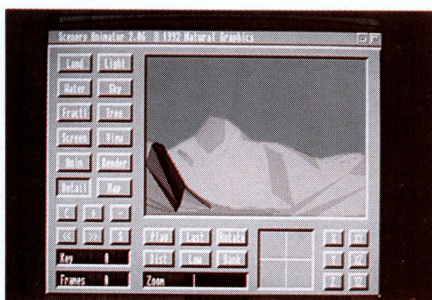
このシリーズは「Scenery」というフリーウェアから始まり，機能が上がるとともに売り物になり価格も上がってきて，最新版は「Scenery Animator ver.2.0」で15,000円ぐらいとなっている。

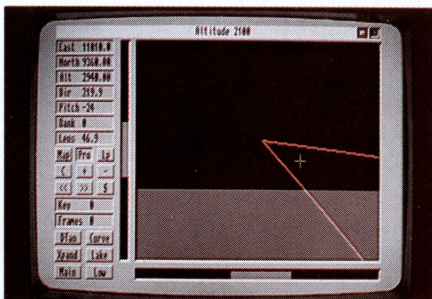
最初のバージョンでは初期値を与えて山の形状を変化させたり，光源の位置を指定できたりしたものの，一定の視点から眺めた風景しか描けなかった。が，雲がつき，視点の位置や向きが変えられるようになるにともなって，フライスルーアニメーションなども作成できるようになったのである。最新版では木も生えてくる。

メイン画面には現在の視点からの風景のプレビューが表示される。視点を変えると，さすがに一瞬のうちにというわけにはいかないが，数秒間程度で再表示してくれる。このおかげで，思いつくままに視点や海面高度などを変更しながら，好みの風景を探すということが可能になっているのだ。また，プレビュー画面上でボックスを指定して部分拡大することもできる。

視点はMAPモードの鳥瞰図で右クリックすれば位置が決まり，左クリックで目標が定まる。このインタフェイスは実にわかりやすく，作業も簡単になっている。ここでは好みの場所をクリックして湖も作れる。

さて，肝心の描画であるが，このソフトは山の輪郭を重ねるようにして，画面を塗っていく。視界などはあまり気にせず，はるか彼方から一番手前まで輪郭を１本１本描いてしまうので，あとで手前の山が重なって見えなくなるような箇所も塗ってしまう。木（２種類）も枝を１本１本伸ばして，葉を１枚１枚つけてというふうに，生真面目に描いていく。そのため時間はかかるが，山の斜面は滑らかに，そして，木々はアップになっても実にリアルに仕上がる。

ただ気に入らないのは，草地と岩肌の境目などがまだらになること。そして，それがはっきりと見えてしまい，かえって汚く感じることである。これは遠くも近くもはっきりと画面に描き込まれていくところにも問題があるようだ。

プレビューをボックスで指定

ボックスで囲んだところが拡大表示される

視点の移動はマウスで

視線の上下も同様

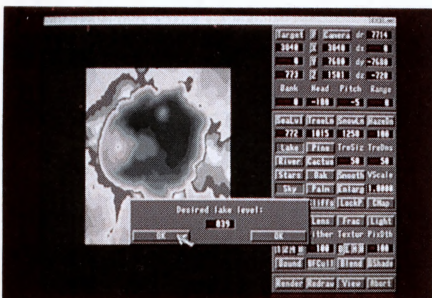
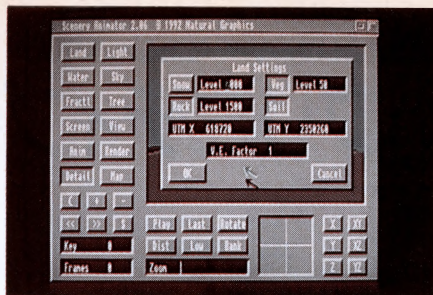
VISTA PRO(Virtual Reality Laboratories)

こちらは「Scenery」シリーズとは逆に，わりと高価なアプリケーションとして出発した。その後，機能を上げてきたにもかかわらず価格は下がってきて，現在は「Scenery Animator ver.2.0」とほぼ同じ値段で売られている。競合するソフトがあると，お互いを意識して高い機能とコストパフォーマンスが実現されるといういい見本であろう。

こちらは地表をポリゴンに変換してレンダリングするという手法をとっている。ポリゴンの大きさは4段階に調整でき，マッピングをすることも可能だが，ポリゴンをいちばん細かくしてもやはりカクカクしてしまう。そこで，一般のポリゴンレンダラーと同様に，グローシェーディングも用意されている。地形の凹凸をほかの3Dレンダリング用オブジェクトデータとして吐き出すこともできる。

また，DōGA CGAシステムでいうところの空気遠近法を使って，遠くやごく近くの地表をぼかして不自然さを解消することも可能だ。4種類の木もポリゴンで描かれるが，演劇の小道具の書き割りのようなものなので，これは立体感があまりない。特にアップになったときなどは貧弱さが目立ってしまうが，その分レンダリングは速い。

海面もやはり分割されたポリゴンで表現されるが，こちらはいい結果を出している。「波あり」にするとポリゴンが小さい凹凸を作り出し，これにグローシェーディングを施すと，なかなかリアルな海面が出来上がるのである。

ペイントツールで木を植える

湖は水面の高さを与えて作成

雪や岩の高度分布は数値を入力

しかし，このソフトの最大の特徴は，地表の高度分布や木，水，建物の分布などをグラフィックデータに書き出せることである。ペイントソフトを使って，地形を変えたり（山を削るなど），特定の地域に木を生やしたりできるのはかなり便利である。うまく使えば，自分が実際に見た風景を再現することや，地図から地形を起こすことも楽にできる。

また，川は始点を指定すれば自動的に作ってくれる。自然の川が流れるように周りの起伏に応じて，低いところへと伸びていくのである。

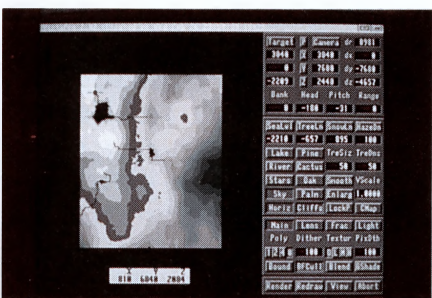
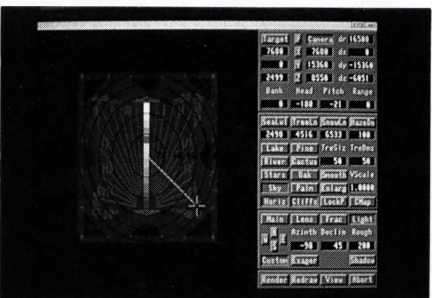
＊　＊　＊

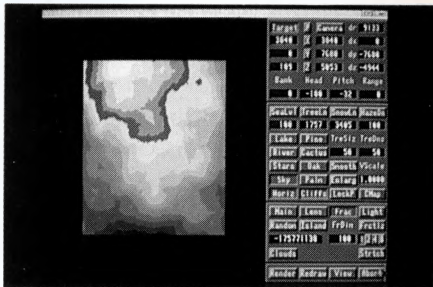
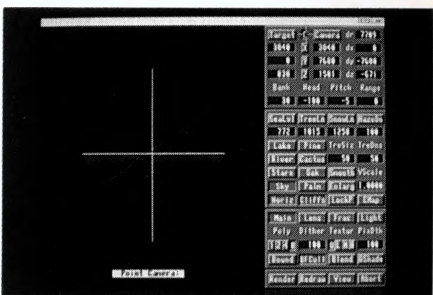
共通する特徴を以下に挙げておこう。

○24ビットカラーで計算できる

○視点の向きや移動パスを指定して，アニメーションを作成できる

○グランドキャニオンやヨセミテ山脈などの地形データが別売で用意されている

最後の1点に関しては，「VISTA」シリーズで地形データに採用されていたDEMファイルが，「Scenery Animator ver.2.0」でも使えるようになった。これによって，地形データの共有が可能になったわけだ。

川は自動的に伸びていく

光源の位置を変更することもできる

輪郭が1本ずつ引かれていく

SEED値を入力すると地形が出来上がる

ワイヤーでプレビューを見る

画像の仕上がり具合は写真を見て判断してもらえばいいが，個人的な感想も述べておこう。リアルさでは「Scenery Animator ver.2.0」，仕上がりの美しさでは「VISTA PRO」に軍配が上がると思う。

「Scenery Animator ver.2.0」は全体的にくっきりしていて写真に近いニュアンス。「VISTA PRO」は雲やモヤ，そして海がボヤッという感じで迫ってきて，どちらかというと絵画的だ。

というわけで，なにか風景をバックに置きたいときなど，「VISTA PRO」を多用している。そのまま使うだけでも十分実用に耐えうる画像だが，1月号で寺尾響子さんがやっていたように3Dレンダリングソフトなどとうまく融合できるとまた面白いと思う。ワイングラスに水が入っていて，湖が真ん中にある地形，あるいは島が浮いているシーンとか，アイデアはいくつかあるのだが，実行には移していない。なにしろ，どちらも計算時間がかかるものなので……。

ぷよぷよびろーんぷるんぷるん

柔らかいプリミティブへの道

Tan Akihiko
丹　明彦

ガラスや金属，大理石……CGの得意とする質感には硬質なものが多い。曲面を使った表現でさえ「柔らかさ」を表すのは容易ではない。ここでは柔らかさの持つ「動き」を加えた造形方法を探ってみよう。

最近のCGの普及ぶりには目を見張るものがあるが，いかにもCGでございといわんばかりの作品も目につく。CGは，珍しがって使う段階から効果を出すために使う段階に入ってきたのだろうとは思うのだが，高級イメージの演出に使われている例は決して少なくない。テレビなんかで見るCGは，どうも綺麗きれいしすぎている。別に悪いことではないし，僕はそういうCGも好きなのだが，それだけがCGってわけじゃあない。

CG非現実性・3つの要因

テレビCMなどでCGを利用しているものを見かけるが，綺麗は綺麗でも絵としては面白味に欠けるものが多い。

1)　形状モデリングの問題

現状では，形状モデリングは多かれ少なかれ手作業である。形状要素(プリミティブ)には，2次曲面やポリゴンをはじめとして多様なものが存在するが，それらを組み合わせるのは基本的に手作業である。

車などの工業製品の表現はCGの得意分野。いわゆるインダストリアルデザインの分野ではCGが大活躍している。

で，逆にCGの苦手としているのが，生き物などの柔らかい物体である。手作業でどれほど上手に作っても，精巧な人形といったイメージを払拭することは難しい。

現在もっとも精巧なモデリング手法のひとつは，現実の物体(たとえば人間)の外形をレーザー光などで計測し，形状データを自動生成してしまうやり方。これにビデオ取り込みのテクスチャマップをかける。

無からすべてを作り出す伝統的なモデリング手法は，人工物を表現できても自然物に対しては限界を露呈するということなのだろうか。そう決めつけてしまうのは早すぎるような気もする。

2)　モーションデザインの問題

一般論として，動きをすべて手作業でデザインするのは間違っていると思う。たとえば，人間の歩行のモーションデザインがものすごく高等な技術だということは，この方面では半ば常識と化している。

これについては百聞は一見にしかずということで，少し前の本誌で正しく花瓶を落として見せてくれた柴田氏の記事を思い起こしていただきたい。簡単な力学法則を入れるだけで，CGの物体は本物そっくりに動き出すのである。

で，前項の柔らかい物体と絡むが，柔らかい物体が柔らかく見える大きな要因は，いうまでもなく，伸びたり縮んだり揺れたり，といった動きがあるためである。この動きのデザインを手作業でやるのは，手間がかかるうえ，自然な動きに見せるのが相当に難しい。人間の目というものは残酷なほどに敏感で，ほんのちょっとした違いで「生きた人間」と「精巧な人形」を区別してしまうのだ。

3)　レンダリングの問題

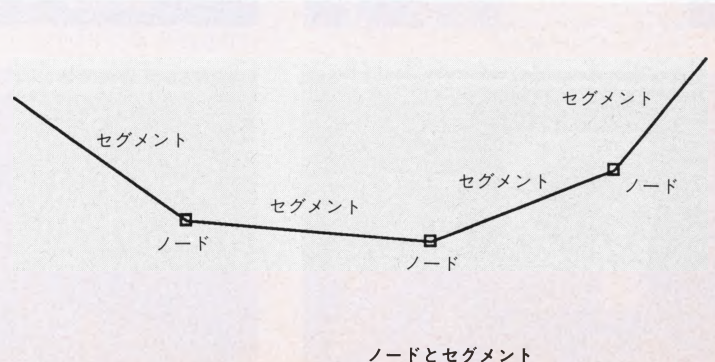
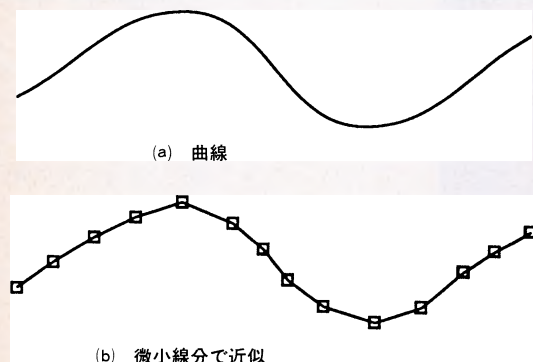
F1マシンなんかのぴかぴかしたボディは，ちょっとしたレンダリングソフトを使えば実写と見紛うばかりの美しさを発揮する。

が，こうしたCG画像には現実感が薄いといわれる。ひとつの大きな理由として，そこが「完璧に透明な空間」であるからというのがある。CGはこの透明な空間が一番安易に表現できるからそうなっているのだが，それがまたCG独特の質感ということになっているようだ。が，CGのわりと最先端に近いところでは，いかにして現実感を出すか，ということに挑戦している。ちょっと言葉が悪いが，CG画像を上手に「汚す」技術を研究しているわけだ。

戦略

今回は柔らかいプリミティブを作るのが大目標だ。そのために，形状データに以下の特徴を盛り込んでみた。

・1枚の膜を図形小片の集まりで表現する
・各小片の情報として座標だけでなく小片

図1　膜の2次元表現

どうしの接続情報を入れる
・人間が与えるのは初期状態と拘束条件
・各小片は外部から(重力など)，またはお互いに(張力による相互作用など)力のやり取りを行う
・各小片について運動方程式を解く
・求めた小片の加速度を微小時間について加算して速度を更新する
・求めた小片の速度を微小時間について加算して小片の位置を更新する
・新しい形状を描画する

以下，これらに対して解説を試みる。

幾何データ構造

今回シミュレートするのは「膜」だが，諸般の事情により2次元空間の中でシミュレートする。そのため，膜は本来は曲面だが，曲線としてモデルを立てる(図1-a)。2次元でうまくいけば，3次元に拡張するのはそれほど困難なことではない。

この曲線を，今回は微小線分の集まりとして近似表現する(図1-b)。3次元に拡張すれば，膜を微小多角形の集まりとして表現するということになる。

さらに詳細に見ると，この曲線は，実際の線を構成する「セグメント」と，隣接するセグメントを接合する点「ノード」からなっている(図1-c)。

曲線には開曲線と閉曲線を用意した(図2)。開曲線は通常の膜を表し，閉曲線はシャボン玉のような閉じた曲面を表す。データ構造上はループを成すか否かで区別している。

また，固定点という概念を設けた(図3)。膜の縁がどこかに固定されて，膜がぶら下がっているという状態を表現する。固定点は移動することがない。主に開曲線の端点での使用を想定している。

シミュレーションの流れ

1) まず，膜の初期状態を決定する(図4)。
2) 各時刻(0.1秒後，0.2秒後というような)ごとに以下を繰り返す。現在時刻をt，時間間隔をΔtとする。
3) 時刻tにおける各ノードおよびセグメントの力学的振る舞いを計算する(図5-a)。
4) 最終的に時刻tに各ノードにかかる合力Fを求め，運動方程式F=maを解き，加速度aを求める(図5-b)。mはノードの質量(膜の質量がノードごとに集中していると仮定した)。
5) 時刻tにおけるノードの速度vの微小変

図2 開曲線と閉曲線

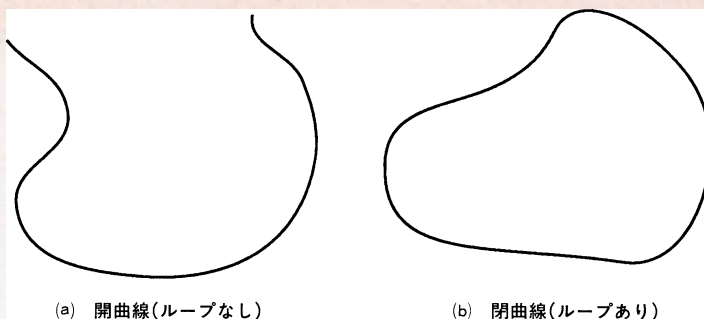



図3 固定点と自由点

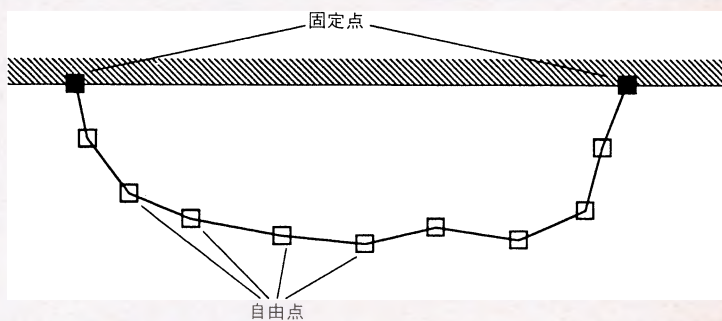



図4 初期状態

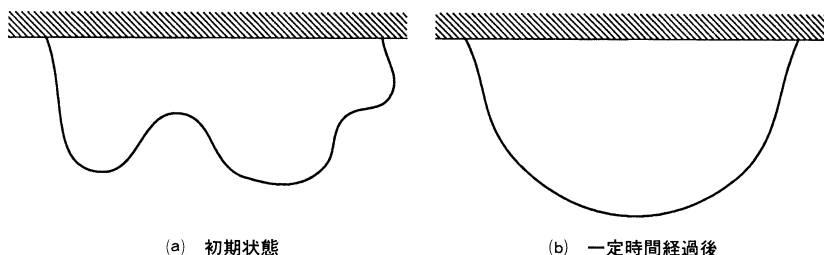
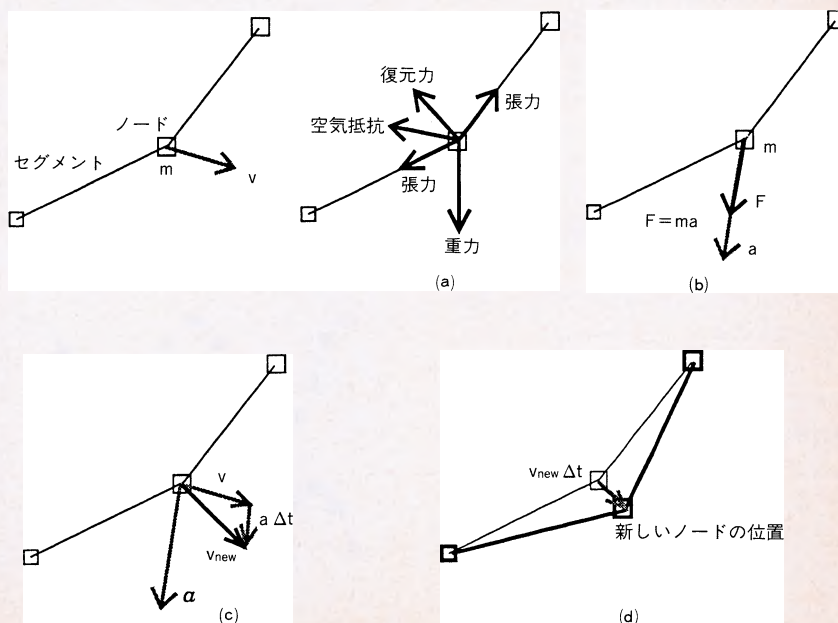



図5 シミュレーションの流れ

化Δvを$a\Delta t$で表す。時刻$(t+\Delta t)$における新しい速度は$(v+a\Delta t)$である(図5-c)。

6) 同様に，ノードの位置の微小変化は$v\Delta t$となる(図5-d)。ノードの位置を更新し，膜を再描画する。

7) 6)において，ノードが画面内に設定した仮想的な壁に衝突した場合は，ノードを壁に反射させる(図6)。具体的には，ノードの速度vの壁に対して垂直な成分を反転させる。

8) 3)に戻る。

発散を防ぐ(時間刻み調整)

上記のような，各時刻ごとに微小変化を計算し，それを加算していく方法は，物理量を時間方向に積分していくのと同じ考え方である。

ただ，このやり方には大きな欠点がある。それは，時間刻みを十分細かく取らなかった場合に計算精度が著しく低下することである。最悪の場合，解は発散して，大デタラメな答えとなる。

速度を積分すると位置を求めることができる，これはすでに説明した。問題は，速度のサンプリング点が極端に粗かった場合。意地悪な例を挙げよう。(図7-a)は真の速度。それを正確に積分すると，(図7-b)のような位置変化になるはずだが，(図7-c)のように時間刻みが粗いと，それを加算した結果は(図7-d)のように，真の値からずるずると離れていってしまう。ちなみに(図7-e)，(図7-f)は時間刻みを細かく取った場合。

いろいろと条件を変えてシミュレーションを行っていて，膜が変な動きをしたら，この時間刻みを疑うとよい。

4つの力

以下はノードおよびセグメントに関連する力として今回考えたものを解説する。

●重力の影響

いわずとしれた引力の法則である。重力加速度をgとすると，ノードには，

$F=mg$

の重力が垂直下方向にかかる(図8)。

●弾力その1(膜の伸縮)

輪ゴムを手で引き伸ばすと，輪ゴムは縮もうとする。あれである。

ここではセグメントに対してバネの性質をあてはめた。セグメントの自然長をlとし，その自然長からΔlだけ伸び縮みすると，セグメントは大きさ，

$F=(k\Delta l)$

の力で元の長さに戻ろうとする(図9)。ここでkはセグメントに固有の係数である。

●弾力その2(曲率保存)

(注意)この項はほかにも増してインチキ臭いので，信用しないように。

たとえば，輪ゴムを握り潰してくちゃくちゃにしたとしても，手を離せば元の形に戻る。これをシミュレートするために，怪しげな理屈を使った。

図6 壁での反射

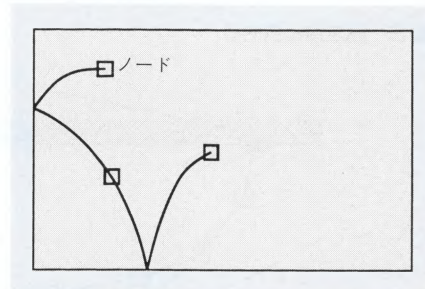
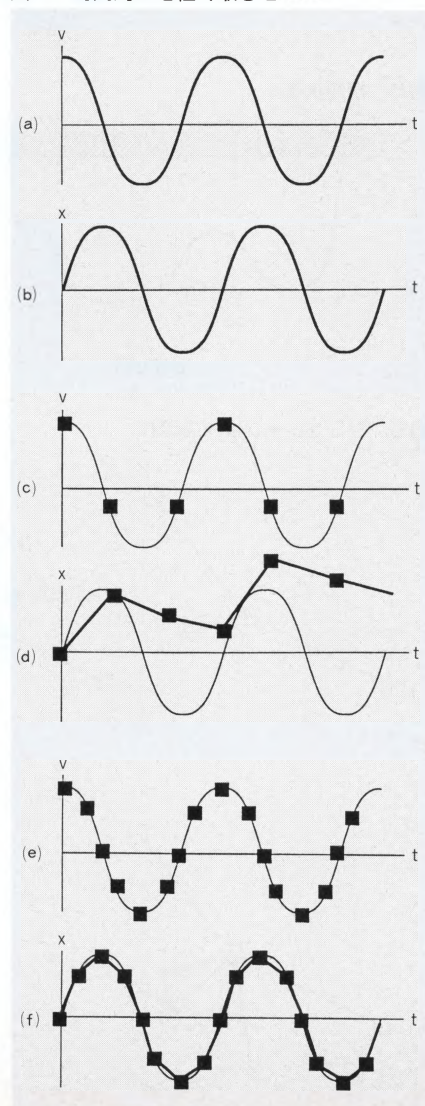



図7 時間刻みを粗く取ると……

まず，ノードおよびそれと隣り合う2本のセグメントの間には角度がつくが，それに「自然な角度」というものが存在する(図10-a)。変形によってこの角度が自然角から離れる(図10-b)と，角度が自然角に戻るようにしようとする力がノードに対して働く(図10-c)。角度の差をΔaとすると，力は，

$F=k\Delta a$

とした。kというのはノードに設定された係数である。

●空気抵抗

さて，引力の法則のもとでは，重いものも軽いものも同じ速度で落ちる。しかし，たとえば鉄球と紙きれを同時に放り出して

図8 重力

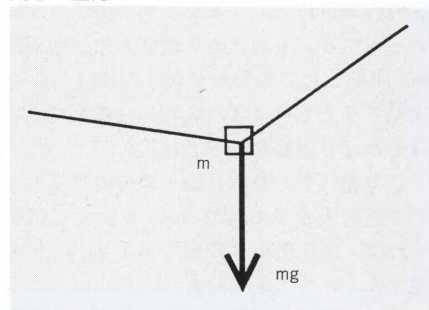



図9 膜の伸縮による弾力

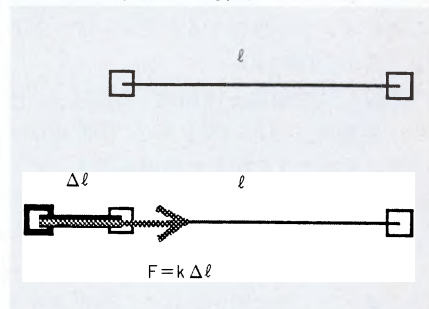
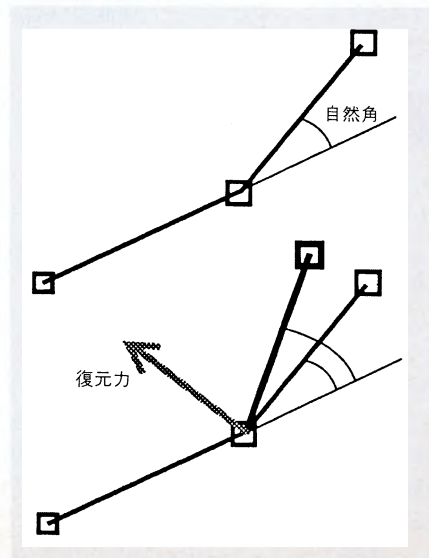



図10 曲率保存

も同時に落下しないという場面では，この法則に理屈ではともかく感覚で納得できないとしても不思議ではない。この原因となっているのが空気抵抗である。空気抵抗のない真空中では，紙切れもすとんと落っこちるのである。なんだか不思議。

空気抵抗にも近似式がある。空気抵抗は空気中を運動する物体の，運動方向とは逆の方向に発生する(図11)。その大きさは速度に比例し，

F＝kv

で表せる。kは例によってノードに設定された係数である。

プログラムの作り方使い方

プログラムはCで書いてあり，いくつかのファイルに分かれている。コンパイル方法はMakefileを参考にしていただきたい。

実行は簡単で，単に，

main ［return］

とすればよろしい。

＊　＊　＊

このプログラムは完全に実験段階のため，データを外部から入力できるなどといった凝ったことをしていない。定数類はソースの中に直接記述してある。

いじるところは，

○const.hの中の物理定数
○const.hの中の時間刻み関連の定数
○main.cの中の関数nextStep()の中身(どの力を考慮に入れるかなどといったことを選択する)
○main.cの中の関数main()の中身(初期状態を設定する)

といったところ。詳しくはソースリスト中のコメントを参照していただきたい。

＊　＊　＊

実例として，

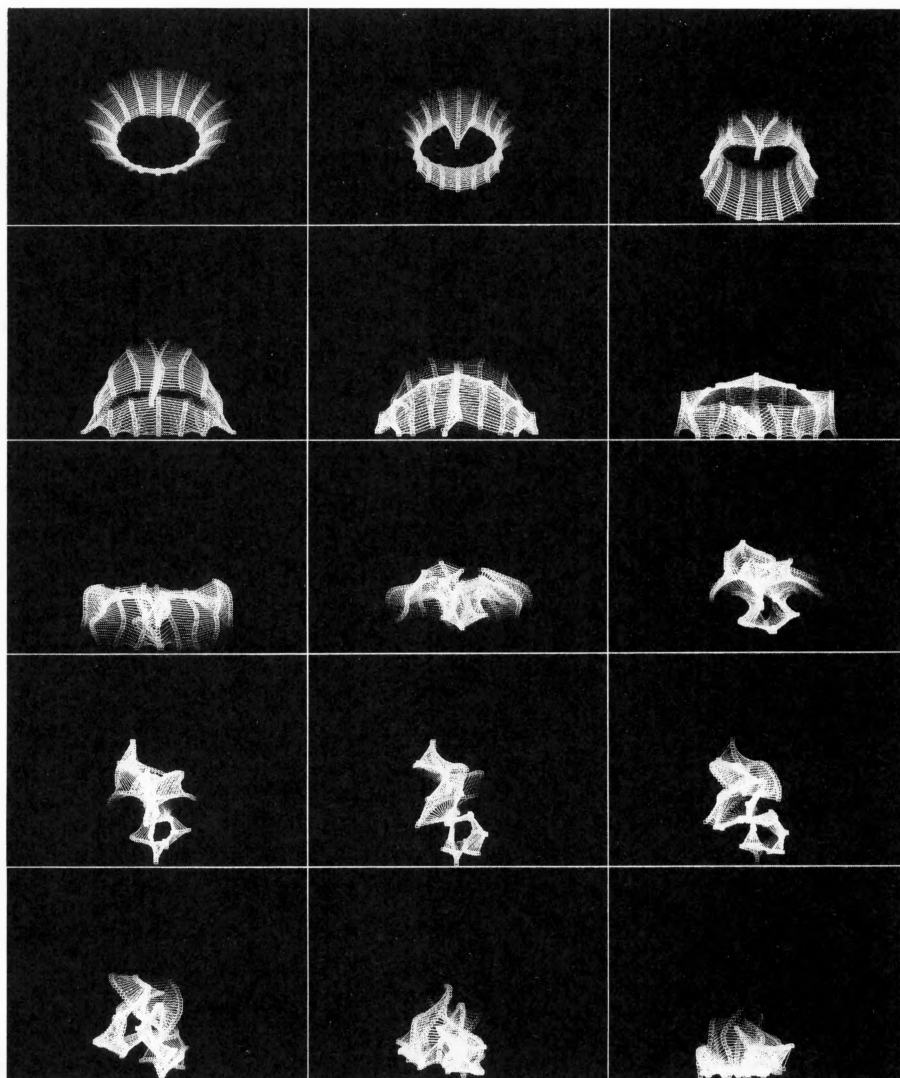
・天井に両端を固定した放物線状の膜
・天井に1端を固定した放物線状の膜
・壁に両端を固定した放物線状の膜
・空中に放った閉じた膜

を挙げておく。

今後の課題または言い訳

今回はちょっと地味だった。目標は本当に遠大で，本記事はその端緒を示したにすぎない。文字どおり，千里の道の一歩状態なわけだ。

ともあれ，弾力のあるプリミティブを表現できる可能性は示せたと思う。

今後の課題は，本来の目的である人体モデリングに必要な機能を追加することだが，ちょっと考えただけでも，

重力による落下と衝突反射の様子

・より正確なシミュレーション(物理定数を導入し，現象もより精密に解く)
・3次元化(「皮膚」)
・ソリッド化(「脂肪」や「筋肉」)
・干渉チェック
・注入/吸引式モデリング(ふくらませたい場所に「脂肪」を注入するようなタイプのモデリング)
・「骨格」「筋肉」によるマクロなモーションデザインと「皮膚」「脂肪」によるミクロなモーションデザインの融合
・衣服の表現

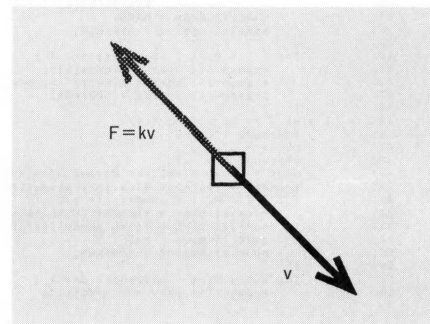
といったところが挙がる。野望は……，えっちな人体モデルを表現できるポテンシャルを持ったプリミティブを実現すること。

これが自然物CGのプラットフォームとして完成するのか，ここで可能性を示唆しただけで終わるのか，僕にもわかっていない。最近このパターンが多い。反省。

終わりに

自動生成というのは面白いものだ。ごく基本的な法則だけをプログラムする。不確定要素が多ければ多いほど楽しい。

種を仕掛ける。そしてあとは計算機任せ。さあいけ，うまく育てよ。うまくいってもいかなくても，どんなものができてくるか考えただけでわくわくするじゃないか。

図11　空気抵抗

リスト1

```
  1: /*
  2:  *     main.c
  3:  *     1992/12     A.Tan
  4:  */
  5:
  6: #include     <math.h>
  7: #include     "primitive.h"
  8: #include     "stage.h"
  9: #include     "const.h"
 10:
 11: void    clearForce();
 12: void    gravity();
 13: void    springSegment();
 14: void    springNode();
 15: void    airRegister();
 16: void    pseudoIntegral();
 17: void    reflection();
 18:
 19: double  xs[16], ys[16], vxs[16], vys[16];
 20: int     fs[16];
 21:
 22: /* 天井に両端を固定した放物線状の膜 */
 23:
 24: void    setup1()
 25: {
 26:     int    i;
 27:     for ( i = 0; i < 16; i++ ) {
 28:         xs[i] = 128 + i * 16;
 29:         ys[i] = 64 + 256*sin(PI*i/15);
 30:         vxs[i] = 0.0;
 31:         vys[i] = 0.0;
 32:         fs[i] = FALSE;
 33:     }
 34:     fs[0] = fs[15] = TRUE;
 35:     createStructure( xs, ys, vxs, vys, fs, 16, FALSE );
 36:     return;
 37: }
 38:
 39: /* 天井に1端を固定した放物線状の膜 */
 40:
 41: void    setup2()
 42: {
 43:     int    i;
 44:     for ( i = 0; i < 16; i++ ) {
 45:         xs[i] = 256;
 46:         ys[i] = 64 + i*16;
 47:         vxs[i] = 0.0;
 48:         vys[i] = 0.0;
 49:         fs[i] = FALSE;
 50:     }
 51:     fs[0] = TRUE;
 52:     createStructure( xs, ys, vxs, vys, fs, 16, FALSE );
 53:     for ( i = 0; i < 16; i++ ) {
 54:         xs[i] = 256 + i * 8;
 55:         ys[i] = 64 + 64*sin(PI*i/15);
 56:         vxs[i] = 0.0;
 57:         vys[i] = 0.0;
 58:         fs[i] = FALSE;
 59:     }
 60:     fs[0] = TRUE;
 61:     setStructure( xs, ys, vxs, vys, fs );
 62:     return;
 63: }
 64:
 65: /* 壁に両端を固定した放物線状の膜 */
 66:
 67: void    setup3()
 68: {
 69:     int    i;
 70:     for ( i = 0; i < 16; i++ ) {
 71:         xs[i] = 64 + 256*sin(PI*i/15);
 72:         ys[i] = 128 + i * 16;
 73:         vxs[i] = 0.0;
 74:         vys[i] = 0.0;
 75:         fs[i] = FALSE;
 76:     }
 77:     fs[0] = fs[15] = TRUE;
 78:     createStructure( xs, ys, vxs, vys, fs, 16, FALSE );
 79:     return;
 80: }
 81:
 82: /* 空中に放った閉じた膜 */
 83:
 84: void    setup4()
 85: {
 86:     int    i:
 87:     for ( i = 0; i < 16; i++ ) {
 88:         xs[i] = 256 - 100*sin(PI*2.0*i/16);
 89:         ys[i] = 128 - 100*cos(PI*2.0*i/16);
 90:         vxs[i] = -50*sin(PI*2.0*i/16);
 91:         vys[i] = -50*cos(PI*2.0*i/16);
 92:         fs[i] = FALSE;
 93:     }
 94:     createStructure( xs, ys, vxs, vys, fs, 16, TRUE );
 95:     for ( i = 0; i < 16; i++ ) {
 96:         xs[i] = 256 - 128*sin(PI*2.0*i/16);
 97:         ys[i] = 256 - 128*cos(PI*2.0*i/16);
 98:         vxs[i] = 0.0;
 99:         vys[i] = 0.0;
100:         fs[i] = FALSE;
101:     }
102:     setStructure( xs, ys, vxs, vys, fs );
103:     return;
104: }
105:
106: void    nextStep()
107: {
108:     /* 初期化 */
109:     clearForce();
110:     /* この下の4つは好きな組み合わせで呼べる */
111:     gravity();
112:     springSegment();
113:     springNode();
114:     airRegister();
115:     /* この下の2つは必ず呼ぶこと */
116:     pseudoIntegral();
117:     reflection();
118:     return;
119: }
120:
121: void    main()
122: {
123:     int    i;
124:
125:     initializeStage();
126:     /*setup1();*/
127:     setup2();
128:     /*setup3();*/
129:     /*setup4();*/
130:     for ( frame = 1; frame <= MAXFRAME + AFTERGLOW; frame++ ) {
131:         drawStructure();
132:         setPalet();
133:         for ( i = 0; i < INTERVAL; i++ ) nextStep();
134:     }
135:     setPaletLast();
136:     disposeStructure();
137: }
```

リスト2

```
  1: /*
  2:  *     primitive.c
  3:  *     1992/12     A.Tan
  4:  */
  5:
  6: #include     <graph.h>
  7: #include     "primitive.h"
  8: #include     "stage.h"
  9: #include     "const.h"
 10:
 11: Segment *segment;
 12: Node    *node;
 13: int     nSegment, nNode, loopMode;
 14:
 15: /*
 16:  *     構造を生成する
 17:  *     引数は座標と速度と固定フラグと数とループモード
 18:  *     内部でsetStructure(), fixStructure()を呼んでいる
 19:  */
 20:
 21: void   createStructure( xs, ys, vxs, vys, fs, n, loop )
 22: double *xs, *ys, *vxs, *vys;
 23: int    *fs, n, loop;
 24: {
 25:     int    i;
 26:     char   *malloc();
 27:
 28:     if ( loop ) {
 29:         loopMode = TRUE;
 30:         nNode = n;
 31:         nSegment = n;
 32:         node = (Node *)malloc( sizeof(Node)*nNode );
 33:         segment = (Segment *)malloc( sizeof(Segment)*nSegment );
 34:         for ( i = 0; i < nNode; i++ ) {
 35:             node[i].prev = &(segment[(i==0)?(nSegment-1):(i-1)]);
 36:             node[i].next = &(segment[i]);
 37:             node[i].mass = MASS;
 38:             node[i].spring = SPRINGN;
 39:         }
 40:         for ( i = 0; i < nSegment; i++ ) {
 41:             segment[i].prev = &(node[i]);
 42:             segment[i].next = &(node[(i==(nSegment-1))?(0):(i+1)]);
 43:             segment[i].spring = SPRINGS;
 44:         }
 45:     } else { /* if ( loop ) */
 46:         loopMode = FALSE;
 47:         nNode = n;
 48:         nSegment = n - 1;
 49:         node = (Node *)malloc( sizeof(Node)*nNode );
 50:         segment = (Segment *)malloc( sizeof(Segment)*nSegment );
 51:         for ( i = 0; i < nNode; i++ ) {
 52:             node[i].prev = ((i==0)?(NULL):&(segment[i-1]));
 53:             node[i].next = ((i==(nNode-1))?(NULL):&(segment[i]));
 54:             node[i].mass = MASS;
 55:             node[i].spring = SPRINGN;
 56:         }
 57:         for ( i = 0; i < nSegment; i++ ) {
 58:             segment[i].prev = &(node[i]);
 59:             segment[i].next = &(node[i+1]);
 60:             segment[i].spring = SPRINGS;
 61:         }
 62:     } /* if ( loop ) */
 63:     setStructure( xs, ys, vxs, vys, fs );
 64:     fixStructure();
 65:     return;
 66: }
 67:
 68: /*
 69:  *     各nodeの座標と速度と固定フラグを設定する
 70:  */
 71:
 72: void    setStructure( xs, ys, vxs, vys, fs )
 73: double  *xs, *ys, *vxs, *vys;
 74: int     *fs;
 75: {
 76:     int i;
 77:
 78:     for ( i = 0; i < nNode; i++ ) {
 79:         node[i].location[0] = xs[i];
 80:         node[i].location[1] = ys[i];
 81:         node[i].velocity[0] = vxs[i];
 82:         node[i].velocity[1] = vys[i];
 83:         node[i].force[0] = 0.0;
 84:         node[i].force[1] = 0.0;
 85:         node[i].fixed = fs[i];
 86:     }
 87:     return;
 88: }
 89:
 90: /*
 91:  *     構造を破棄する
 92:  */
 93:
 94: void    disposeStructure()
 95: {
 96:     void    free();
 97:
 98:     free( node );
 99:     free( segment );
100:
101:     return;
102: }
103:
104: /*
105:  *     セグメントの方向ベクトルを求める
106:  */
107:
108: void    segmentVector( v, s )
109: VECTOR  v;
110: Segment *s;
111: {
112:     subtractVector( v, ((Node*)(s->next))->location,
113:                        ((Node*)(s->prev))->location );
114:     return;
115: }
116:
```


▶X68000を買ってからだいぶたちましたが，X-BASICは使いこなすのが大変です。X-BASICは難しいですね。
渡辺　晃一(19)長野県

```
117: /*
118:  *      構造を固定する
119:  *      現在の構造が安定であるように設定する
120:  */
121: 
122: void    fixStructure()
123: {
124:     int         i;
125:     VECTOR      tv, tv1;
126: 
127:     if ( loopMode ) {
128:         for ( i = 0; i < nNode; i++ ) {
129:             segmentVector( tv, (Segment*)((node[i].next)) );
130:             normalizeVector( tv, tv );
131:             segmentVector( tv1, (Segment*)((node[i].prev)) );
132:             normalizeVector( tv1, tv1 );
133:             node[i].angle = relativeTheta( tv, tv1 );
134:         }
135:         for ( i = 0; i < nSegment; i++ ) {
136:             segmentVector( tv, &segment[i] );
137:             segment[i].length = lengthVector( tv );
138:         }
139:     } else { /* if ( loopMode ) */
140:         if ( node[0].fixed ) {
141:             segmentVector( tv, (Segment*)((node[0].next)) );
142:             normalizeVector( tv, tv );
143:             node[0].angle = absoluteTheta( tv );
144:         } else {
145:             node[0].angle = 0.0;
146:         }
147:         for ( i = 1; i < nNode - 1; i++ ) {
148:             segmentVector( tv, (Segment*)((node[i].next)) );
149:             normalizeVector( tv, tv );
150:             segmentVector( tv1, (Segment*)((node[i].prev)) );
151:             normalizeVector( tv1, tv1 );
152:             node[i].angle = relativeTheta( tv, tv1 );
153:         }
154:         if ( node[ nNode - 1 ].fixed ) {
155:             segmentVector( tv, (Segment*)((node[nNode-1].prev)) );
156:             normalizeVector( tv, tv );
157:             node[ nNode - 1 ].angle = absoluteTheta( tv );
158:         } else {
159:             node[ nNode - 1 ].angle = 0.0;
160:         }
161:         for ( i = 0; i < nSegment; i++ ) {
162:             segmentVector( tv, &segment[i] );
163:             segment[i].length = lengthVector( tv );
164:         }
165:     } /* if ( loopMode ) */
166:     return;
167: }
168: 
169: /*
170:  *      構造を描画する
171:  */
172: 
173: void    drawStructure()
174: {
175:     int   i;
176:     int   x1, y1, x2, y2;
177: 
178:     for ( i = 0; i < nSegment; i++ ) {
179:         x1 = (int)(((Node*)(segment[i].prev))->location[0]);
180:         y1 = (int)(((Node*)(segment[i].prev))->location[1]);
181:         x2 = (int)(((Node*)(segment[i].next))->location[0]);
182:         y2 = (int)(((Node*)(segment[i].next))->location[1]);
183:         line( x1, y1, x2, y2, frame, 0xFFFF );
184:     }
185:     for ( i = 0; i < nNode; i++ ) {
186:         x1 = (int)(node[i].location[0]) - 3;
187:         y1 = (int)(node[i].location[1]) - 3;
188:         x2 = x1 + 6;
189:         y2 = y1 + 6;
190:         box( x1, y1, x2, y2, frame, 0xFFFF );
191:     }
192:     return;
193: }
```

リスト3

```
  1: /*
  2:  *     vector.c
  3:  *     1992/12   A.Tan
  4:  */
  5: 
  6: #include    <stdio.h>
  7: #include    <math.h>
  8: #include    "vector.h"
  9: 
 10: void    printVector( v )
 11: VECTOR  v;
 12: {
 13:     int  i;
 14:     for ( i = 0; i < DIMENSION_VECTOR; i++ ) printf( "%f ", v[i] );
 15:     printf( "¥n" );
 16:     return;
 17: }
 18: 
 19: void    addVector( d, s1, s2 )    /* d = s1 + s2 */
 20: VECTOR  d, s1, s2;
 21: {
 22:     int i;
 23:     for ( i = 0; i < DIMENSION_VECTOR; i++ ) d[i] = s1[i] + s2[i];
 24:     return;
 25: }
 26: 
 27: void    subtractVector( d, s1, s2 )    /* d = s1 - s2 */
 28: VECTOR  d, s1, s2;
 29: {
 30:     int i;
 31:     for ( i = 0; i < DIMENSION_VECTOR; i++ ) d[i] = s1[i] - s2[i];
 32:     return;
 33: }
 34: 
 35: void    multiplyVector( d, s1, s2 )    /* d = s1 s2 */
 36: VECTOR  d, s2;
 37: double  s1;
 38: {
 39:     int i;
 40:     for ( i = 0; i < DIMENSION_VECTOR; i++ ) d[i] = s1 * s2[i];
 41:     return;
 42: }
 43: 
 44: double  dotVector( s1, s2 )    /* s1・s2 */
 45: VECTOR  s1, s2;
 46: {
 47:     int     i;
 48:     double  d = 0.0;
 49:     for ( i = 0; i < DIMENSION_VECTOR; i++ ) d += s1[i] * s2[i];
 50:     return ( d );
 51: }
 52: 
 53: double    lengthVector( s )    /* |s1| */
 54: VECTOR    s;
 55: {
 56:     int    i;
 57:     double d = 0.0;
 58:     for ( i = 0; i < DIMENSION_VECTOR; i++ ) d += s[i] * s[i];
 59:     return ( sqrt( d ) );
 60: }
 61: 
 62: void    normalizeVector( d, s )    /* d = s/|s| (|d|=1) */
 63: VECTOR  d, s;
 64: {
 65:     int     i;
 66:     double  l = lengthVector( s );
 67:     if ( l == 0.0 )
 68:         for ( i = 0; i < DIMENSION_VECTOR; i++ ) d[i] = 0.0;
 69:     else
 70:         for ( i = 0; i < DIMENSION_VECTOR; i++ ) d[i] = s[i] / l;
 71:     return;
 72: }
 73: 
 74: double   theta( s, c )
 75: double   s, c;
 76: {
 77:     if ( c == 0.0 ) {
 78:         if ( s > 0.0 ) return ( PI/2.0 );
 79:         return ( -PI/2.0 );
 80:     }
 81:     if ( s >= 0.0 ) {
 82:         if ( c > 0.0 ) return ( atan( s/c ) );
 83:         return ( PI - atan( -s/c ) );
 84:     } else {    /* s < 0.0 */
 85:         if ( c < 0.0 ) return ( atan( s/c ) - PI );
 86:         return ( -atan( -s/c ) );
 87:     }
 88: }
 89: 
 90: double    absoluteTheta( v )
 91: VECTOR    v;
 92: {
 93:     VECTOR    tv;
 94:     normalizeVector( tv, v );
 95:     return ( theta( tv[1], tv[0] ) );
 96: }
 97: 
 98: double    relativeTheta( v1, v2 )
 99: VECTOR    v1, v2;
100: {
101:     VECTOR    tv1, tv2;
102:     normalizeVector( tv1, v1 );
103:     normalizeVector( tv2, v2 );
104:     return ( theta( tv1[1], tv1[0] ) - theta( tv2[1], tv2[0] ) );
105: }
```

リスト4

```
  1: /*
  2:  *     stage.c
  3:  *     1992/12    A.Tan
  4:  */
  5: 
  6: #include    <basic0.h>
  7: #include    <graph.h>
  8: #include    "stage.h"
  9: 
 10: int     frame;     /* 何フレームめか */
 11: 
 12: /*
 13:  *     舞台を初期化する
 14:  */
 15: 
 16: void    initializeStage()
 17: {
 18:     int     i;
 19:     /* 512×512ドット256色モード */
 20:     screen( 1, 2, 1, 1 );
 21:     /* パレットを初期化する */
 22:     for ( i = 0; i < 256; i++ ) palet( i, RGB(0,0,0) );
 23:     return;
 24: }
 25: 
 26: /*
 27:  *     各フレームのパレットを設定する
 28:  *     残像が残るような効果がある
 29:  */
 30: 
 31: void    setPalet()
 32: {
 33:     int     i, j, k;
 34:     palet( frame-AFTERGLOW, RGB(0,0,0) );
 35:     for ( i = 1; i <= AFTERGLOW; i++ ) {
 36:         k = frame - AFTERGLOW + i;
 37:         if ( k < 1 ) continue;
 38:         if ( k > MAXFRAME ) break;
 39:         j = GRAYSCALE - AFTERGLOW + i - 1;
 40:         palet( k, RGB(j,j,j) );
 41:     }
 42:     return;
 43: }
 44: 
 45: /*
 46:  *     シミュレーション最終結果のパレット処理
 47:  */
 48: 
 49: void    setPaletLast()
 50: {
 51:     int     i;
 52:     for ( i = 0; i < MAXFRAME; i++ )
 53:         palet( i, RGB(0,0,0) );
 54:     palet( MAXFRAME, RGB(31,31,31) );
 55:     return;
 56: }
```

▶以前MZを所有していましたが，一時PC-9801に移っていたため，作る人から使う人になってしまいました。X68000も購入して1年ほどたちますが，ほとんど使用していません。Oh!Xを読みながら，作る人へと戻ろうと思っています。　　　澤田　均(26)神奈川県

リスト5

```
  1: /*
  2:  *     simulation.c
  3:  *     1992/12    A.Tan
  4:  */
  5:
  6: #include    <math.h>
  7: #include    "primitive.h"
  8: #include    "stage.h"
  9: #include    "const.h"
 10:
 11: VECTOR    g = { 0.0, G };     /* 重力ベクトル */
 12:
 13: /*
 14:  *     nodeに加わる力をクリアする
 15:  *     各時刻の計算の始めに呼び出す
 16:  */
 17:
 18: void    clearForce()
 19: {
 20:     int    i;
 21:     for ( i = 0; i < nNode; i++ ) {
 22:         node[i].force[0] = 0.0;
 23:         node[i].force[1] = 0.0;
 24:     }
 25:     return;
 26: }
 27:
 28: /*
 29:  *     各時刻ごとのnodeの新しい速度と位置を求める
 30:  */
 31:
 32: void    pseudoIntegral()
 33: {
 34:     int    i;
 35:     for ( i = 0; i < nNode; i++ ) {
 36:         if ( node[i].fixed ) {
 37:             node[i].velocity[0] = 0.0;
 38:             node[i].velocity[1] = 0.0;
 39:         } else {
 40:             node[i].velocity[0] += TIMESTEP * (node[i].force[0]) / node[i].
mass;
 41:             node[i].velocity[1] += TIMESTEP * (node[i].force[1]) / node[i].
mass;
 42:             node[i].location[0] += TIMESTEP * (node[i].velocity[0]);
 43:             node[i].location[1] += TIMESTEP * (node[i].velocity[1]);
 44:         }
 45:     }
 46:     return;
 47: }
 48:
 49: /*
 50:  *     壁にぶつかったときに反射する
 51:  */
 52:
 53: void    reflection()
 54: {
 55:     int    i;
 56:     for ( i = 0; i < nNode; i++ ) {
 57:         if ( node[i].location[0] < BOUNDX1 ) {
 58:             node[i].location[0] = BOUNDX1;
 59:             node[i].velocity[0] *= -REFLECTIONX;
 60:         }
 61:         if ( node[i].location[0] > BOUNDX2 ) {
 62:             node[i].location[0] = BOUNDX2;
 63:             node[i].velocity[0] *= -REFLECTIONX;
 64:         }
 65:         if ( node[i].location[1] < BOUNDY1 ) {
 66:             node[i].location[1] = BOUNDY1;
 67:             node[i].velocity[1] *= -REFLECTIONY;
 68:         }
 69:         if ( node[i].location[1] > BOUNDY2 ) {
 70:             node[i].location[1] = BOUNDY2;
 71:             node[i].velocity[1] *= -REFLECTIONY;
 72:         }
 73:     }
 74:     return;
 75: }
 76:
 77: /*
 78:  *     重力を計算する
 79:  */
 80:
 81: void    gravity()
 82: {
 83:     int       i;
 84:     VECTOR    tv;
 85:     for ( i = 0; i < nNode; i++ ) {
 86:         multiplyVector( tv, node[i].mass, g );
 87:         addVector( node[i].force, node[i].force, tv );
 88:     }
 89:     return;
 90: }
 91:
 92: /*
 93:  *     segmentの伸縮に伴う張力を求める
 94:  */
 95:
 96: void    springSegment()
 97: {
 98:     int       i;
 99:     Node      *p, *n;
100:     VECTOR    v1, v2;
101:     double    dl;
102:     for ( i = 0; i < nSegment; i++ ) {
103:         p = segment[i].prev;
104:         n = segment[i].next;
105:         subtractVector( v1, n->location, p->location );
106:         dl = lengthVector( v1 ) - segment[i].length;
107:         normalizeVector( v2, v1 );
108:         multiplyVector( v1, (segment[i].spring) * dl, v2 );
109:         addVector( p->force, p->force, v1 );
110:         subtractVector( n->force, n->force, v1 );
111:     }
112:     return;
113: }
114:
115: /*
116:  *     nodeの角度の変化に伴う復元力を求める
117:  */
118:
119: void    springNode()
120: {
121:     int      i;
122:     VECTOR   tv1, tv2, tv3;
123:     double   a, b;
124:     if ( loopMode ) {
125:         for ( i = 0; i < nNode; i++ ) {
126:             segmentVector( tv1, (Segment*)((node[i].next)) );
127:             normalizeVector( tv1, tv1 );
128:             segmentVector( tv2, (Segment*)((node[i].prev)) );
129:             normalizeVector( tv2, tv2 );
130:             a = relativeTheta( tv1, tv2 );
131:             if ( a == node[i].angle ) continue;
132:             if ( a > node[i].angle ) {
133:                 b = (a + PI)/2.0;
134:             } else {
135:                 b = (a - PI)/2.0;
136:             }
137:             tv1[0] = -tv2[1]; tv1[1] = tv2[0];
138:             multiplyVector( tv3, node[i].spring * cos( b ), tv2 );
139:             addVector( node[i].force, node[i].force, tv3 );
140:             multiplyVector( tv3, node[i].spring * sin( b ), tv1 );
141:             addVector( node[i].force, node[i].force, tv3 );
142:         }
143:     } else {    /* if ( loopMode ) */
144:         for ( i = 1; i < nNode - 1; i++ ) {
145:             segmentVector( tv1, (Segment*)((node[i].next)) );
146:             normalizeVector( tv1, tv1 );
147:             segmentVector( tv2, (Segment*)((node[i].prev)) );
148:             normalizeVector( tv2, tv2 );
149:             node[i].angle = relativeTheta( tv1, tv2 );
150:             if ( a == node[i].angle ) continue;
151:             if ( a > node[i].angle ) {
152:                 b = (a + PI)/2.0;
153:             } else {
154:                 b = (a - PI)/2.0;
155:             }
156:             tv1[0] = -tv2[1]; tv1[1] = tv2[0];
157:             multiplyVector( tv3, node[i].spring * cos( b ), tv2 );
158:             addVector( node[i].force, node[i].force, tv3 );
159:             multiplyVector( tv3, node[i].spring * sin( b ), tv1 );
160:             addVector( node[i].force, node[i].force, tv3 );
161:         }
162:     } /* if ( loopMode ) */
163:     return;
164: }
165:
166: /*
167:  *     空気抵抗を求める
168:  */
169:
170: void    airRegister()
171: {
172:     int      i;
173:     VECTOR   tv;
174:     double   k;
175:
176:     for ( i = 0; i < nNode; i++ ) {
177:         k = -REGISTER * lengthVector( node[i].velocity );
178:         normalizeVector( tv, node[i].velocity );
179:         multiplyVector( tv, k, tv );
180:         addVector( node[i].force, node[i].force, tv );
181:     }
182:     return;
183: }
```

リスト6

```
 1: /*
 2:  *     const.h
 3:  *     1992/12    A.Tan
 4:  */
 5:
 6: #ifndef    CONST_H
 7: #define    CONST_H
 8:
 9: /* 一般定数 */
10:
11: #ifndef    NULL
12: #define    NULL    ((void *)0)
13: #endif    /* NULL */
14:
15: #ifndef    TRUE
16: #define    TRUE    1
17: #endif    /* TRUE */
18:
19: #ifndef    FALSE
20: #define    FALSE    0
21: #endif    /* FALSE */
22:
23: /* 時間刻み定数 */
24:
25: #define    TIMESTEP    (0.1)    /* 時間刻み */
26: #define    INTERVAL    5        /* 表示間隔 */
27:
28: /* 物理定数 */
29:
30: #define    G              (8.0)  /* 重力加速度 */
31: #define    REFLECTIONX    (1.0)  /* はね返り係数 */
32: #define    REFLECTIONY    (1.0)  /* はね返り係数 */
33: #define    REGISTER       (0.01) /* 空気抵抗係数 */
34: #define    MASS           (1.0)  /* nodeの質量 */
35: #define    SPRINGN        (1.0)  /* nodeのばね定数 */
36: #define    SPRINGS        (8.0)  /* segmentのばね定数 */
37:
38: #endif    /* CONST_H */
```

リスト7

```
 1: /*
 2:  *     primitive.h
 3:  *     1992/12    A.Tan
 4:  */
 5:
 6: #ifndef    PRIMITIVE_H
 7: #define    PRIMITIVE_H
 8:
 9: #include    "vector.h"
10:
11: typedef struct Node {
12:     void      *prev;       /* 前の Segment へのポインタ */
13:     void      *next;       /* 次の Segment へのポインタ */
14:     VECTOR    location;    /* 位置 */
15:     VECTOR    velocity;    /* 速度 */
16:     VECTOR    force;       /* 力 */
17:     int       fixed;       /* 固定されているかどうか */
18:     double    angle;       /* 適切な(相対)角度 */
19:     double    mass;        /* 質量 */
20:     double    spring;      /* バネ定数(角度方向) */
21: } Node;
22:
23: typedef struct Segment {
24:     void      *prev;       /* 前の Node へのポインタ */
25:     void      *next;       /* 次の Node へのポインタ */
26:     double    length;      /* 適切な長さ */
27:     double    spring;      /* バネ定数(長さ方向) */
28: } Segment;
29:
30: extern Segment *segment;
31: extern Node    *node;
32: extern int     nSegment, nNode, loopMode;
33:
34: void   createStructure( double*, double*, double*, double*, int*, int, int
);
35: void   setStructure( double*, double*, double*, double*, int* );
36: void   disposeStructure();
37: void   drawStructure();
38: void   segmentVector( VECTOR, Segment* );
39: void   fixStructure();
40:
41: #endif    /* PRIMITIVE_H */
```

▶私はピザが嫌いです。やはり，お好み焼きにこそ日本人の心があると思うのです。

堀川　英雄(23)大分県

リスト8

```
 1: /*
 2:  *     stage.h
 3:  *     1992/12     A.Tan
 4:  */
 5:
 6: #ifndef    STAGE_H
 7: #define    STAGE_H
 8:
 9: #define    GRAYSCALE  32
10: #define    AFTERGLOW  24
11: #define    MAXFRAME   255
12:
13: #define    BOUNDX1    32
14: #define    BOUNDY1    32
15: #define    BOUNDX2    480
16: #define    BOUNDY2    480
17:
18: #define    RGB(R,G,B)    (((G)<<11)|((R)<<6)|((B)<<1))
19:
20: extern int frame;
21:
22: void    initializeStage();
23: void    setPalet();
24: void    setPaletLast();
25:
26: #endif     /* STAGE_H */
```

リスト9

```
 1: /*
 2:  *     vector.c
 3:  *     1992/12     A.Tan
 4:  */
 5:
 6: #ifndef    VECTOR_H
 7: #define    VECTOR_H
 8:
 9: /* デフォルトでは2次元 */
10: #ifndef    DIMENSION_VECTOR
11: #define    DIMENSION_VECTOR    2
12: #endif     /* DIMENSION_VECTOR */
13:
14: /* VECTOR型 */
15: typedef    double    VECTOR[ DIMENSION_VECTOR ];
16:
17: /* 関数宣言 */
18: void    printVector( VECTOR );
19: void    addVector( VECTOR, VECTOR, VECTOR );
20: void    subtractVector( VECTOR, VECTOR, VECTOR );
21: void    multiplyVector( VECTOR, double, VECTOR );
22: double  dotVector( VECTOR, VECTOR );
23: double  lengthVector( VECTOR );
24: void    normalizeVector( VECTOR, VECTOR );
25: double  theta( double, double );
26: double  absoluteTheta( VECTOR );
27: double  relativeTheta( VECTOR, VECTOR );
28:
29: #endif     /* VECTOR_H */
```

リスト10

```
 1: /*
 2:  *     const.h
 3:  *     1992/12     A.Tan
 4:  */
 5:
 6: #ifndef    CONST_H
 7: #define    CONST_H
 8:
 9: /* 一般定数 */
10:
11: #ifndef    NULL
12: #define    NULL     ((void *)0)
13: #endif     /* NULL */
14:
15: #ifndef    TRUE
16: #define    TRUE     1
17: #endif     /* TRUE */
18:
19: #ifndef    FALSE
20: #define    FALSE    0
21: #endif     /* FALSE */
22:
23: /* 時間刻み定数 */
24:
25: #define    TIMESTEP     (0.1)     /* 時間刻み */
26: #define    INTERVAL     5         /* 表示間隔 */
27:
28: /* 物理定数 */
29:
30: #define    G            (8.0)     /* 重力加速度 */
31: #define    REFLECTIONX  (1.0)     /* はね返り係数 */
32: #define    REFLECTIONY  (1.0)     /* はね返り係数 */
33: #define    REGISTER     (0.01)    /* 空気抵抗係数 */
34: #define    MASS         (1.0)     /* nodeの質量 */
35: #define    SPRINGN      (1.0)     /* nodeのばね定数 */
36: #define    SPRINGS      (8.0)     /* segmentのばね定数 */
37:
38: #endif     /* CONST_H */
```

リスト11

```
 1: /*
 2:  *     const.h
 3:  *     1992/12     A.Tan
 4:  */
 5:
 6: #ifndef    CONST_H
 7: #define    CONST_H
 8:
 9: /* 一般定数 */
10:
11: #ifndef    NULL
12: #define    NULL     ((void *)0)
13: #endif     /* NULL */
14:
15: #ifndef    TRUE
16: #define    TRUE     1
17: #endif     /* TRUE */
18:
19: #ifndef    FALSE
20: #define    FALSE    0
21: #endif     /* FALSE */
22:
23: /* 時間刻み定数 */
24:
25: #define    TIMESTEP     (0.1)     /* 時間刻み */
26: #define    INTERVAL     5         /* 表示間隔 */
27:
28: /* 物理定数 */
29:
30: #define    G            (8.0)     /* 重力加速度 */
31: #define    REFLECTIONX  (1.0)     /* はね返り係数 */
32: #define    REFLECTIONY  (1.0)     /* はね返り係数 */
33: #define    REGISTER     (0.01)    /* 空気抵抗係数 */
34: #define    MASS         (1.0)     /* nodeの質量 */
35: #define    SPRINGN      (1.0)     /* nodeのばね定数 */
36: #define    SPRINGS      (8.0)     /* segmentのばね定数 */
37:
38: #endif     /* CONST_H */
```

リスト12

```
 1: /*
 2:  *     const.h
 3:  *     1992/12     A.Tan
 4:  */
 5:
 6: #ifndef    CONST_H
 7: #define    CONST_H
 8:
 9: /* 一般定数 */
10:
11: #ifndef    NULL
12: #define    NULL     ((void *)0)
13: #endif     /* NULL */
14:
15: #ifndef    TRUE
16: #define    TRUE     1
17: #endif     /* TRUE */
18:
19: #ifndef    FALSE
20: #define    FALSE    0
21: #endif     /* FALSE */
22:
23: /* 時間刻み定数 */
24:
25: #define    TIMESTEP     (0.1)     /* 時間刻み */
26: #define    INTERVAL     5         /* 表示間隔 */
27:
28: /* 物理定数 */
29:
30: #define    G            (1.0)     /* 重力加速度 */
31: #define    REFLECTIONX  (1.0)     /* はね返り係数 */
32: #define    REFLECTIONY  (1.0)     /* はね返り係数 */
33: #define    REGISTER     (1.0)     /* 空気抵抗係数 */
34: #define    MASS         (1.0)     /* nodeの質量 */
35: #define    SPRINGN      (8.0)     /* nodeのばね定数 */
36: #define    SPRINGS      (8.0)     /* segmentのばね定数 */
37:
38: #endif     /* CONST_H */
```

リスト13

```
 1: /*
 2:  *     const.h
 3:  *     1992/12     A.Tan
 4:  */
 5:
 6: #ifndef    CONST_H
 7: #define    CONST_H
 8:
 9: /* 一般定数 */
10:
11: #ifndef    NULL
12: #define    NULL     ((void *)0)
13: #endif     /* NULL */
14:
15: #ifndef    TRUE
16: #define    TRUE     1
17: #endif     /* TRUE */
18:
19: #ifndef    FALSE
20: #define    FALSE    0
21: #endif     /* FALSE */
22:
23: /* 時間刻み定数 */
24:
25: #define    TIMESTEP     (0.1)     /* 時間刻み */
26: #define    INTERVAL     1         /* 表示間隔 */
27:
28: /* 物理定数 */
29:
30: #define    G            (8.0)     /* 重力加速度 */
31: #define    REFLECTIONX  (1.0)     /* はね返り係数 */
32: #define    REFLECTIONY  (1.0)     /* はね返り係数 */
33: #define    REGISTER     (0.01)    /* 空気抵抗係数 */
34: #define    MASS         (1.0)     /* nodeの質量 */
35: #define    SPRINGN      (4.0)     /* nodeのばね定数 */
36: #define    SPRINGS      (4.0)     /* segmentのばね定数 */
37:
38: #endif     /* CONST_H */
```

リスト14 MAKEFILE

```
 1: all: main.x
 2:
 3: main.x: main.o simulation.o primitive.o stage.o vector2.o
 4:      gcc main.o simulation.o primitive.o stage.o vector2.o -lbas -lfloatfnc
 5:
 6: main.o: main.c primitive.h vector.h const.h
 7:      gcc -O -c main.c
 8:
 9: simulation.o: simulation.c primitive.h stage.h const.h
10:      gcc -O -c simulation.c
11:
12: primitive.o: primitive.c primitive.h vector.h const.h
13:      gcc -O -c primitive.c
14:
15: stage.o: stage.c stage.h
16:      gcc -O -c stage.c
17:
18: vector2.o: vector.c vector.h
19:      gcc -DDIMENSION_VECTOR=2 -O -c vector.c -o vector2.o
```

▶いまさらウィザードリィにはまっています。現在，私のNinjaはレベル90です。まだ，やっている人なんているんでしょうかね。そんな人の近況を知りたいなあ。

南谷　豊一郎(23)大阪府

モデリングデータのコンバート

CAD_CNV.BAS

Hamazaki Masaya 浜崎　正哉

今回，3D物体をエディットする[illegible]AD.Xが出力するデータをMAGICにコンバートするプ[illegible]ます。これで，面倒なモデリングも楽々，簡単に行える，[illegible]う。

モデラがない！

1991年5月号で発表されたMAGICですが，ゲームなどのアプリケーションは発表されても，環境整備についてはかなり遅れをとっています。まず，なんといっても3D物体をエディットするためのモデラが発表されていません。

僕の制作したSIONIIでは，一応未発表であるモデラ（御木氏制作）を使用しましたが，まだ完成してなく，機能的にもそこそこのものでした。僕自身は，この未発表のモデラである程度満足していたし，ほしけりゃ自分たちでなんとかするだろう，とたかをくくっていました。

それでも，アンケートハガキを読んでいると，「さっさとモデラを発表せんかい」というハガキが目につくこともありました。「やっぱりなんとかしなくちゃならんかな」とモデラを自作する元気のない僕は，既存のものからコンバートするという方向で考え始めていました。

そこで目についたのが，1992年7月号の付録ディスク，DōGA CGAシステムに収録されていたCAD.Xでした。使い勝手はともかく，DōGA CGAシステムのモデラということもあり，機能も豊富だし，データの蓄積もあります。

今回は，このモデリングデータを，MAGICのデータ形式にコンバートするプログラムを制作してみました。

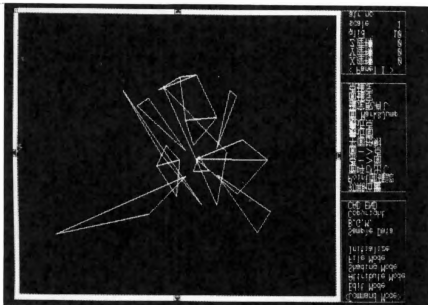



写真1　3Dオブジェクト

データ形式

それでは，MAGIC, CAD.Xでどのようなデータ形式なのか見てみましょう。

まず，写真1のような物体があったとします。リスト1がCAD.Xの定義データ，リスト2がMAGICの定義データとなります。MAGICでは，3D物体データを定義する場合，頂点個数があって，頂点座標（X，Y，Z座標）が続き，頂点のつながりを示す線分データの個数，どの頂点を結ぶかを示す線分データで表されています。

次にターゲットとなっているCAD.Xが出力する＊.SUF形式のデータを見てみましょう（リスト2）。これは，写真1のモデルをCAD.Xでモデリングし，そのままファイル出力したものです。なにやらコマンドが並び，3つごとにきりがよく数字データが並んでいる部分に気づくでしょう。

そして，この3つごとに並んでいる数字が，取り出さなくてはならない頂点データとなります。次に，MAGICで定義されているように，それらの頂点を結ぶ線分データがあればコンバートを楽に行えます。

しかし，リスト2を見てわかるとおり，いまいった頂点を結ぶ線分データらしきものは見つかりません。線分データがないとなると，結局コンバートの際にデータを作成することになります。

CAD.Xでは，基本的に面（ポリゴン）単位でモデリングデータを扱っています。そのポリゴンを定義しているのが，データ部分の直前にある，

prim poly (

という命令です。先ほど述べたとおり，この命令のあとに続いている数字が頂点データです。これは，3角形であれば3個分，4角形であれば4個分，N角形であればN個分の頂点が並ぶことになります。そして，ポリゴンの区切りは，“)”というキャラクタによって行われています。

要するに線分データは，N角形のポリゴンを定義している頂点を，

0-1，1-2，2-3，……，(N-1)-N，N-0

という具合に結べばいいことになります（図1）。

以上で，コンバート作業の基本方針がたったわけですが，もうひとつ手を加えてやらなくてはならないところがあります。それは，DōGAとMAGICの座標系の違いです。といっても図2のように，座標軸の入れ替わりと方向の違いだけですから，それほど面倒なものではありません。

どういったものができたか

以上のように方針を立てて制作したプログラムが，リスト3です。使い方は，プログラムをBASIC上で実行していただければわかると思いますので，ここではパスさせてもらいます。

プログラムの流れを軽く見ていくと，

1)　ポリゴン定義命令文字列の検索
2)　ポリゴンを定義している頂点データの取り出し
3)　頂点を結ぶ線分データの作成
4)　データ終了まで1)～3)を繰り返す
5)　重複点，重複線の削除
6)　指定されたファイルモードでコンバートデータを出力

1)は，行頭に“p”の文字があると“prim poly (”命令である，ということにしています。2)の頂点データを取り出すところは，定義終了キャラクタ“)”がくるまで座標をpoint_data()の配列に格納して，point_cntにその頂点個数を格納していきます。3)の線分データの作成は，前項のデータ形式で説明したとおりのことを行っています。

ここまでの作業＋座標軸変換で，一応MAGICで表示可能なデータ形式にすることができます。しかし，このままではMAGICにとって無駄となる重複点，重複線が残るのです。たとえば，図3のようにポリ

ゴンが隣接して定義されている場合を考えます。CAD.Xで出力したデータでは2つのポリゴンで定義されることになります。すると，図3の2点と，それに付随している線分が重複してしまうのです。そこで，もうひとつ5)の作業が必要となるのです。

この重複点，重複線を除く作業の順番としては，

1) 重複点の削除&線分番号の書き換え
2) 重複点削除に伴う空きエリアの整理&線分番号の書き換え
3) 重複線の削除

となっています。言葉で説明するよりも，変換の様子を図4に書いておきましたので，詳しく知りたい方はそちらを参考にしてください。あまりにも幼稚なアルゴリズムで恐縮なのですが，一応まともに動いているのでごかんべんを。

ファイル出力

現在，このプログラムでは出力するデータ形式として，

1) X-BASIC
2) ASSEMBLER
3) *_M. DAT

の3つに対応しています。1)では行番号なしのX-BASIC形式のリスト，2)ではアセンブラでそのまま使えるデータ形式です。ここでは，頭にMAGICの物体定義コマンドである$0Cを埋め込むようになっていますが，拡張モード時に必要となる色データは，組み込まれませんので注意してください。

最後の「*_M.DAT」というのは，コンバートしたデータを生のままファイルに落

リスト1

```
obj suf ship {
/* cad line color = 15 */
atr no
prim poly  (   -260      -40      -20
                 40      -80       40
                170      -20      140 )

prim poly  (   -260       40      -20
                170       20      140
                 40       80       40 )

prim poly  (     40        0       20
                100      -40      -50
                100       40      -50 )

prim poly  (    100       40      -50
                  0       20     -120
                  0      -20     -120
                100      -40      -50 )

prim poly  (      0       20     -120
               -180        0      -40
                  0      -20     -120 )

prim poly  (      0       20       60
                140       80      120
                180      150       40 )

prim poly  (    180      150       40
                120      100        0
                  0       20       60 )

prim poly  (    180     -150       40
                140      -80      120
                  0      -20       60 )

prim poly  (      0      -20       60
                120     -100        0
                180     -150       40 )

prim poly  (    100      -40       20
                -20     -120      -60
                  0     -100      -80 )

prim poly  (      0      100      -80
                -20      120      -60
                100       40       20 )

prim poly  (    100       40       80
                150      160      160
               -220      200      220 )

prim poly  (    100      -40       80
                150     -160      160
               -220     -200      220 )

prim poly  (     80        0       40
                100      -20       70
                100       20       70 )
}
```

リスト2

```
        dc.w    35
        dc.w    -40,20,260
        dc.w    -80,-40,-40
        dc.w    -20,-140,-170
        dc.w    40,20,260
        dc.w    20,-140,-170
        dc.w    80,-40,-40
        dc.w    0,-20,-40
        dc.w    -40,50,-100
        dc.w    40,50,-100
        dc.w    20,120,0
        dc.w    -20,120,0
        dc.w    0,40,180
        dc.w    20,-60,0
        dc.w    80,-120,-140
        dc.w    150,-40,-180
        dc.w    100,0,-120
        dc.w    -150,-40,-180
        dc.w    -80,-120,-140
        dc.w    -20,-60,0
        dc.w    -100,0,-120
        dc.w    -40,-20,-100
        dc.w    -120,60,20
        dc.w    -100,80,0
        dc.w    100,80,0
        dc.w    120,60,20
        dc.w    40,-20,-100
        dc.w    40,-80,-100
        dc.w    160,-160,-150
        dc.w    200,-220,220
        dc.w    -40,-80,-100
        dc.w    -160,-160,-150
        dc.w    -200,-220,220
        dc.w    0,-40,-80
        dc.w    -20,-70,-100
        dc.w    20,-70,-100
        dc.w    39
        dc.w    0,1
        dc.w    1,2
        dc.w    0,2
        dc.w    3,4
        dc.w    4,5
        dc.w    3,5
        dc.w    6,7
        dc.w    7,8
        dc.w    6,8
        dc.w    8,9
        dc.w    9,10
        dc.w    10,7
        dc.w    9,11
        dc.w    11,10
        dc.w    12,13
        dc.w    13,14
        dc.w    12,14
        dc.w    14,15
        dc.w    15,12
        dc.w    16,17
        dc.w    17,18
        dc.w    16,18
        dc.w    18,19
        dc.w    19,16
        dc.w    20,21
        dc.w    21,22
        dc.w    20,22
        dc.w    23,24
        dc.w    24,25
        dc.w    23,25
        dc.w    26,27
        dc.w    27,28
        dc.w    26,28
        dc.w    29,30
        dc.w    30,31
        dc.w    29,31
        dc.w    32,33
        dc.w    33,34
        dc.w    32,34
```

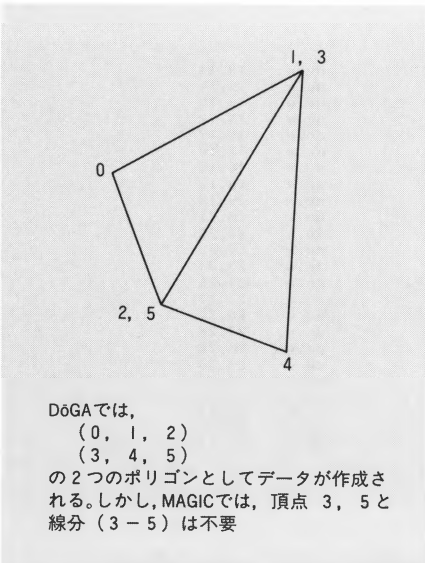
図1 線分データの構成

図2 座標系の違い

としたものです。どのように利用するかは，あなたしだい。とりあえず，コンバートした「＊_M.DAT」形式のファイルにあるデータをMAGICで簡易表示するプログラムをリスト4に用意しておきました（要MAGIC.FNC)。プログラムを実行すると，読み込むファイル名を聞いてきますので，「_M.DAT」を省略したファイル名を入力してください。データの読み込みが終わると，3D物体がクルクルと回り，“＋”“－”キーで物体の位置を調節できます。

なお，このプログラムではエラーチェックなど行っていませんので，大きなデータを表示させたい場合は，object(),read_bufの配列要素数を大きくしてください。

使用上の注意

まず，コンバート元のデータは，CAD.Xから直接出力したものしか受けつけません。お試しディスクの入っているモデリングデータも，一度CAD.Xに読み込んで別のファイル名で出力するようにしてください。問題となっているのは，頂点データの区切りにタブコードが使われている場合ですので，タブコードがなければそのままコンバートできます。とりあえず，一度CAD.Xに読み込んで出力したデータでは，お試しディスクにあったデータも問題なくコンバートできたので，大丈夫でしょう。

次に，バッファとして確保している配列，point_data( ),line_data( ),object_data( )の要素数もコンバートしようとする元データによって，大きくする必要があります。もしも，大きくした場合は変換できる総頂点数（point_max)，総線分数（line_max）も変更しておきましょう。

目安としては，総頂点数を基準にして，

総頂点数×4＝総線分数

総頂点数＋総線分数＝総オブジェクト配列要素数

で大丈夫だと思います。

また，大きなデータ（頂点数1000とか）をコンバートしようと考えている人は，必ずこのプログラムをコンパイルして使用してください。でないと，途方もない時間がかかります。

どう使うかはあなたしだい

機能の割には，結構大きなプログラムとなってしまいました。まあ，モデラのプログラムを打ち込む手間，作る手間を考えれば楽なものですからね。

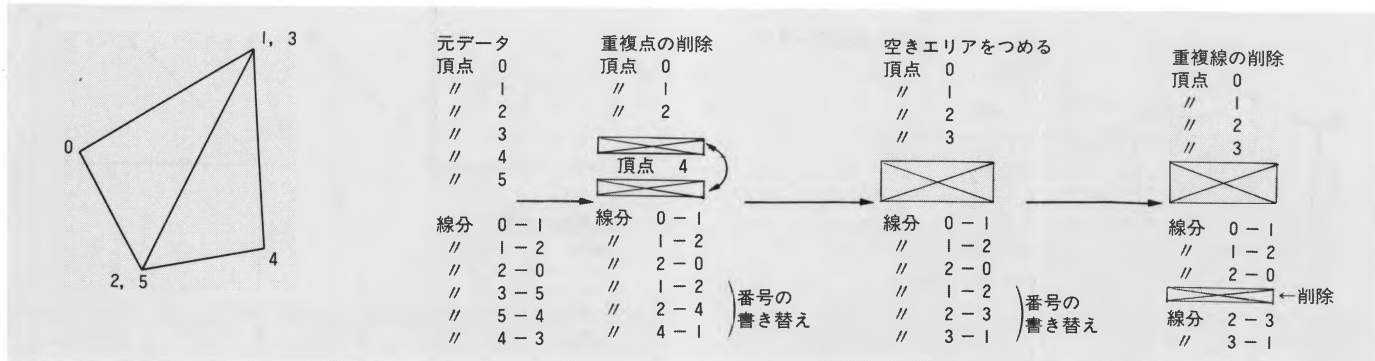
それにしても，サポートがあまりないとはいえMAGICを使った投稿がないのは，寂しいかぎりです。以前，中矢士朗さんからの投稿が2本ぐらいあっただけですからね。掲載されるレベルとしては，ちょっともの足りない部分もありましたが，同じものを使っている人として非常に頼もしい仲間意識を感じたことを覚えています。

また，高速な3Dパッケージという特性を生かしたツールとして，いろいろな用途が考えられるはずです（レイトレソフトの簡易プレビュアなんかとかね)。MAGICは使い込めば使い込むほど，いろいろな可能性が見えて結構奥が深いですよ（別に3Dだから奥行があるということではありません)。がんばって使ってみましょう。

図3　隣接しているポリゴン

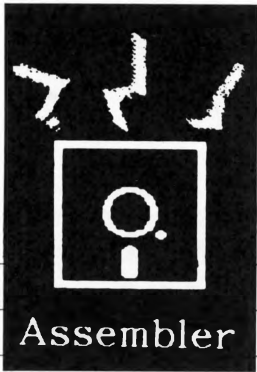
DōGAでは,
(0, 1, 2)
(3, 4, 5)
の2つのポリゴンとしてデータが作成される。しかし, MAGICでは, 頂点 3, 5と線分（3－5）は不要

DōGAお試しディスクにあったデータをコンバートしてみたもの。配列の要素数は,
point_data(4000, 2)
line_data(12000, 1)
object_max(16000)
で実行

図4　重複点，重複線削除の様子

```
10 /*
20 /* DoGA CAD データをMAGICのデータにコンバート
30 /*    CAD_CNV.BAS
40 /*
50   str read_buf[255]   /* リードバッファ
60   int i,j,k,l,fp,err
70   int point_cnt=0      /* 頂点数
80   int point_ex         /* 重複点を除いたあとの頂点数
90   int line_cnt=0       /* 線分数
100  int line_ex          /* 重複線を除いたあとの線分数
110  str file_name        /* コンバートするファイル名
120  str dec_str          /* 10進文字列のワーク
130  int cnv_f            /* コンバート開始フラグ
140  int pointer          /* 文字列ポインタ
150  int output_mode      /* 出力データ形式
160  int data_over=0      /* データオーバーフローチェック
170  int file_over=0      /* ファイル読み込みチェック
180  int point_max=51     /* 変換できる総頂点数
190  int line_max=201     /* 変換できる総線分数
200  dim int point_data(50,2) /*頂点データ
210  dim int line_data(200,1) /*線分データ
220  dim int object(250)       /*生成するオブジェクト格納配列
230 /*
240 /* メインルーチン
250 /*
260     input "READ FILE NAME:",file_name
270     print "OUTPUT SOURCE"
280     print "1) X-BASIC"
290     print "2) ASSEMBLER"
300     print "3) *_M.DAT"
310     repeat
320       input "SELECT MODE:",output_mode
330     until output_mode<4 and output_mode>0
340     fp=fopen(file_name+".suf","r")
350     err=fp
360       while err<>-1 and data_over=0 and file_over=0
370       err=freads(read_buf,fp)        /* 1行読み込み
380         if read_buf<>"" then {
390             cnv_f=0
400             if left$(read_buf,1)="p" then cnv_f=100
410 /*
420                 while cnv_f<>0 and err<>-1
430                      poly_cnv()         /* 1ポリゴン変換
440                 endwhile
450         }
460           file_over=feof(fp)
470       endwhile
480      fclose(fp)
490 /*
500      if data_over=0 and file_over<>0 then {
510        print "重複点の削除&線分番号の書き換え"
520         double_point()       /* 重複点の削除
530        print "重複点削除に伴う空きエリアの整理&線分番号の書き換え"
540         space_clr()          /* 重複添削に伴う空きエリアの整理
550        print "重複線の削除"
560         double_line()        /* 重複線の削除
570        print "オブジェクトデータ作成"
580         object_make()        /* オブジェクトデータの作成
590          switch output_mode /* データの出力
600            case 1:write_basic():break
610            case 2:write_as():break
620            case 3:write_dat():break
630          endswitch
640         print "総頂点数:";point_ex
650         print "総線分数:";line_ex
660       } else {
670        beep
680        if data_over<>0 then {
690            print "データが多すぎて変換できませんでした"
700           } else {
710           if err<>0 then {
720            print "ファイルアクセスに失敗しました"
730           } }
740     }
750     end
760 /*
770 /* 1ポリゴンのラインデータをコンバート
780 /*
790 func poly_cnv()
800   int point_buf
810       point_buf=point_cnt
820       print read_buf
830         pointer=strcspn(read_buf,"(")  /* パラメータエリアの検索
840         pointer=pointer+2
850         space_skip()
860         if mid$(read_buf,pointer,1)<>"" then {
870            str_dec()
880         }
890              while right$(read_buf,1)<>")" and data_over=0
900                err=freads(read_buf,fp)    /* 1行読み込み
910                 pointer=1
920                 space_skip()
930                 print read buf
940                 str_dec()
950              endwhile
960           connect_make(point_buf)      /* 線分データの作成
970      cnv_f=0
980 endfunc
990 /*
1000 /* 線分データの作成
1010 /*
1020 func connect_make(start_p)
1030   int p_cnt
1040     p_cnt=point_cnt-start_p      /* いくつ頂点があったか計算
1050     line_cnt=line_cnt+p_cnt      /* 線分個数の更新
1060     if line_cnt<line_max then {
1070       for i=0 to p_cnt-2
1080         line_data(i+start_p,0)=i+start_p
1090         line_data(i+start_p,1)=i+start_p+1
1100       next
1110         line_data(point_cnt-1,0)=start_p   /* 最後の線分の処理
1120         line_data(point_cnt-1,1)=point_cnt-1
1130     } else { data_over=100 }
1140 endfunc
1150 /*
1160 /* 文字列を数値化する(1行分)
1170 /*
1180 func str_dec()
1190   int x,y,z
1200   int answer=0
1210     if point_cnt<point_max then {
1220       for i=0 to 2
1230          dec_get()
1240          point_data(point_cnt,i)=atoi(dec_str)
1250       next
1260     point_cnt=point_cnt+1
1270       } else { data_over=100 }
1280 endfunc
1290 /*
1300 /* 数値文字列の取り出し
1310 /*
1320 func dec_get()
1330   str chk_str
1340   int end_f=0
1350     dec_str=""
1360     while end_f=0
1370         chk_str=mid$(read_buf,pointer,1)
1380         if chk_str<>" " and chk_str<>"" then {
1390                dec_str=dec_str+chk_str
1400                pointer=pointer+1
1410             } else {
1420                end_f=100
1430          }
1440     endwhile
1450     space_skip()
1460 endfunc
1470 /*
1480 /* read_buf中のスペースをスキップ
1490 /*
1500 func space_skip()
1510   int end_f=0
1520   str chk
1530       while end_f=0
1540          chk=mid$(read_buf,pointer,1)
1550          if chk=" " then {
1560              pointer=pointer+1
1570            } else {
1580              end_f=100
1590          }
1600       endwhile
1610 endfunc
1620 /*
1630 /* 重複点の削除
1640 /*
1650 func double_point()
1660    point_ex=point_cnt
1670      for i=0 to point_cnt-2
1680        for j=i+1 to point_cnt-1
1690         if point_data(i,0)<>65536 then{    /*すでに消された点をはじく
1700          if point_data(i,0)=point_data(j,0) then {
1710            if point_data(i,1)=point_data(j,1) then {
1720              if point_data(i,2)=point_data(j,2) then {
1730                point_ex=point_ex-1
1740                point_erase(i,j)
1750          } } } }
1760        next
1770      next
1780 endfunc
1790 /*
1800 /* 重複点の削除&それに伴うライン番号の変更
1810 /*
1820 func point_erase(p_num,d_num)
1830   int k
1840    for k=0 to 2                  /* 頂点の削除
1850      point_data(d_num,k)=65536
1860    next
1870      for k=0 to line_cnt-1      /* ライン番号の書き換え
1880        if line_data(k,0)=d_num then line_data(k,0)=p_num
1890        if line_data(k,1)=d_num then line_data(k,1)=p_num
1900      next
1910 endfunc
1920 /*
1930 /* point_erase()で生じた空きエリアをつめる
1940 /* それに伴うライン番号の変更
1950 /*
1960 func space_clr()
1970   int p=0,pw,end_f
1980    while p<point_ex
1990       end_f=100
2000       if point_data(p,0)=65536 then { pw=p+1:end_f=0
2010          while end_f=0              /* 空きエリアを飛ばす
2020              if point_data(pw,0)<>65536 then{
2030                 for i=0 to 2        /* データの入れ替え
2040                   point_data(p,i)=point_data(pw,i)
2050                   point_data(pw,i)=65536
2060                 next
2070                 for i=0 to line_cnt-1 /* ライン番号の書き換え
2080                    if line_data(i,0)=pw then line_data(i,0)=p
2090                    if line_data(i,1)=pw then line_data(i,1)=p
2100                 next
2110                 end_f=100
2120              }
2130              pw=pw+1
2140          endwhile
2150       }
2160       p=p+1
2170    endwhile
2180 endfunc
2190 /*
2200 /* 重複線の削除
2210 /*
2220 func double_line()
2230   int l1,l2
2240     line_ex=line_cnt
2250      for i=0 to line_cnt-2
2260        l1=line_data(i,0)
2270        l2=line_data(i,1)
2280        if l1<>65536 then {
2290          for j=i+1 to line_cnt-1
2300            if line_data(j,0)=l1 then {
2310              if line_data(j,1)=l2 then {
2320                 line_data(j,0)=65536
2330                 line_data(j,1)=65536
2340                 line_ex=line_ex-1
2350            } } else {
2360              if line_data(j,1)=l1 then {
```

```
2370              if line_data(j,0)=12 then {
2380                 line_data(j,0)=65536
2390                 line_data(j,1)=65536
2400                 line_ex=line_ex-1
2410           } } }
2420           next
2430        }
2440      next
2450 endfunc
2460 /*
2470 /* オブジェクトデータの生成
2480 /*
2490 func object_make()
2500   int num=0
2510    object(num)=point_ex         /* 頂点個数の格納
2520    num=num+1:i=0
2530     while i<point_ex  /* 頂点データのコピー&座標変換
2540       object(num)=point_data(i,1)
2550       object(num+1)=-point_data(i,2)
2560       object(num+2)=-point_data(i,0)
2570       num=num+3
2580     i=i+1
2590     endwhile
2600    object(num)=line_ex  /* 線分個数の格納
2610    num=num+1:i=0
2620     while i<line_cnt
2630       if line_data(i,0)<>65536 then {
2640          object(num)=line_data(i,0):num=num+1
2650          object(num)=line_data(i,1):num=num+1
2660       }
2670       i=i+1
2680     endwhile
2690 endfunc
2700 /*
2710 /* X-BASICで使える形式でファイル出力
2720 /*
2730 func write_basic()
2740   str write_name,put_str
2750   int fp,num=0,all_data
2760     all_data=point_ex*3+line_ex*2+1 /* データの総計
2770     write_name=strupr(file_name)+"_M.BAS"
2780       print "ファイル名:";write_name;"でセーブします。"
2790       fp=fopen(write_name,"c")      /* ファイルの新規作成
2800         put_str="dim int "+file_name+"("+itoa(all_data)
2810         fwrites(put_str+") = {",fp)
2820         cr_put(fp)
2830         fwrites("       ",fp)
2840         fwrites("+"+itoa(object(num))+", ",fp) /* 頂点個数の出力
2850         cr_put(fp):num=num+1
2860         for i=0 to point_ex-1
2870           fwrites("         ",fp)
2880           for j=0 to 2
2890             put_str=""
2900             if object(num)>=0 then put_str="+"
2910             put_str=put_str+itoa(object(num))
2920             fwrites(put_str+", ",fp)
2930             num=num+1
2940           next
2950         cr_put(fp)
2960         next
2970 /*
2980         fwrites("       ",fp)
2990         fwrites("+"+itoa(object(num))+", ",fp)         /* 線分個数の出力
3000         cr_put(fp):num=num+1
3010         for i=0 to line_ex-1
3020           fwrites("         ",fp)
3030           for j=0 to 1
3040             fwrites("+"+itoa(object(num)),fp)
3050             if num<>all_data then {
3060               fwrites(", ",fp) }
3070             num=num+1
3080           next
3090         cr_put(fp)
3100         next
3110         fputc('}',fp):cr_put(fp):fputc(&H1A,fp)
3120       fclose(fp)
3130 endfunc
3140 /*
3150 /* AS.Xで使える形式でファイル出力
3160 /*
3170 func write_as()
3180   str write_name,dc_str
3190   int fp,num=0
3200     dc_str = chr$(9)+"dc.w"+chr$(9)
3210     write_name=strupr(file_name)+"_M.S"
3220       print "ファイル名:";write_name;"でセーブします。"
3230       fp=fopen(write_name,"c")      /* ファイルの新規作成
3240         fwrites("* "+file_name,fp)
3250         cr_put(fp)
3260         fwrites(dc_str+"$0C",fp)   /* MAGICオブジェクト定義コマンド
3270         cr_put(fp)
3280         fwrites(dc_str+itoa(object(num)),fp) /* 頂点個数の出力
3290         cr_put(fp):num=num+1
3300         for i=0 to point_ex-1
3310           fwrites(dc_str,fp)
3320           for j=0 to 2
3330             fwrites(itoa(object(num)),fp)
3340             if j<>2 then fputc(',',fp)
3350             num=num+1
3360           next
3370         cr_put(fp)
3380         next
3390 /*
3400         fwrites(dc_str+itoa(object(num)),fp) /* 線分個数の出力
3410         cr_put(fp):num=num+1
3420         for i=0 to line_ex-1
3430           fwrites(dc_str,fp)
3440           for j=0 to 1
3450             fwrites(itoa(object(num)),fp)
3460             if j<>1 then fputc(',',fp)
3470             num=num+1
3480           next
3490         cr_put(fp)
3500         next
3510         fputc(&H1A,fp)
3520       fclose(fp)
3530 endfunc
3540 /*
3550 /* ベタ形式でファイル出力
3560 /*
3570 func write_dat()
3580   str write_name,dc_str
3590   int save_data
3600     write_name=strupr(file_name)+"_M.DAT"
3610       print "ファイル名:";write_name;"でセーブします。"
3620       fp=fopen(write_name,"c")      /* ファイルの新規作成
3630       save_data=point_ex*3+line_ex*2+2
3640       fwrite(object,save_data,fp)
3650 endfunc
3660 /*
3670 /* 改行コードの書き込み
3680 /*
3690 func cr_put(fp)
3700     fputc(&HD,fp)
3710     fputc(&HA,fp)
3720 endfunc
3730 /*
```

リスト4 MV.BAS

```
 10 /*
 20 /* *_M.DATの簡易ビュア(MV.BAS)
 30 /*
 40  int i,j,fp
 50  int all_point,all_line
 60  int head=0,z=800,end_f=0
 70  str k
 80  dim int object(200)
 90  dim int read_buf(200)
100    all_init()
110    data_read()
120      while end_f=0
130         head=head+1
140         if head=360 then head=0
150         k=inkey$(0)
160         if k="+" then z=z+100
170         if k="-" then z=z-100
180         MAGIC_CHRPUT(head,z)
190      endwhile
200    end
210 /*
220 /* データを読み込む
230 /*
240 func data_read()
250  str file_name,read_name
260  int point_cnt,line_cnt,i,num=0
270     input "READ FILE NAME(*_M.DAT):",file_name
280     read_name=file_name+"_M.DAT"
290     fp=fopen(read_name,"r")
300       fread(read_buf,1,fp)      /*頂点個数の読み込み
310       object(0)=read_buf(0)
320       all_point=object(0)
330       num=num+1
340       point_cnt=read_buf(0)*3  /*頂点個数の計算
350       fread(read_buf,point_cnt,fp)
360         for i=0 to point_cnt-1    /*頂点データの転送
370           object(num)=read_buf(i)
380           num=num+1
390         next
400 /*
410       fread(read_buf,1,fp)     /*線分個数の読み込み
420       object(num)=read_buf(0)
430       all_line=read_buf(0)
440       num=num+1
450       line_cnt=read_buf(0)*2  /*線分個数の計算
460       fread(read_buf,line_cnt,fp)
470         for i=0 to line_cnt-1    /*頂点データの転送
480           object(num)=read_buf(i)
490           num=num+1
500         next
510       print "総頂点数:";all_point
520       print "総線分数:";all_line
530 endfunc
540 /*
550 /* MAGICの初期化
560 /*
570 func all_init()
580       magic_flush()           /*データバッファのクリア
590       magic_init()            /*3Dワークの初期化
600       magic_screen(1)         /*画面モード512×512
610       for i=0 to 8            /*3Dパラメータクリア
620         magic_para(i,0)
630       next
640       magic_color(255)        /*描画色のセット
650       mouse(4)                /*マウスの初期化
660       mouse(2)                /*マウスカーソルの消去
670       screen 1,2,1,1
680       console ,,0
690 endfunc
700 /*
710 /* キャラクターデータ定義
720 /*
730 func MAGIC_CHRPUT(head,z)
740       magic_obj=magic_putbuf(1,object)
750       magic_seek(1,magic_obj,0)
760       magic_data(1)
770       magic_para(2,z)
780       magic_para(6,head)
790       magic_para(7,-25)
800       magic_pers()       /* 3D-2D変換
810       magic_disp()       /* オブジェクト表示
820 endfunc
830 /*
```

バックグラウンド処理

Murata Toshiyuki 村田 敏幸

先月に続き，Human68k ver 2.0の拡張機能を取り上げます。今回は，いくつかの処理を同時に行うためのバックグラウンド処理機能について解説しましょう。この機能を使ったプログラムを具体例に，作成方法とその常駐，実行のやり方を考えてみます

今回は，複数プログラムの並列実行を可能とするバックグラウンド処理機能を使ったプログラムの作り方を中心に，Human68k ver 2.0のプロセス/メモリ管理まわりの機能をみていく。例によって，個々のDOSコールの呼び出し方などについてはあまり触れるつもりはないので，適宜『プログラマーズマニュアル』を参照してもらいたい。

バックグラウンド処理機能の概要

いうまでもないことだが，複数のプログラムが並行して走るとはいっても，CPUがひとつしかない以上，ある瞬間に実行されるプログラムはただひとつだけだ。しかし，プログラムをちょっと実行してはべつのプログラムをまたちょっと実行する，というように制御を十分短い間隔で切り替えてやれば，複数のプログラムが並列動作しているようにみえる。これがいわゆるマルチタスキングの基本だ。

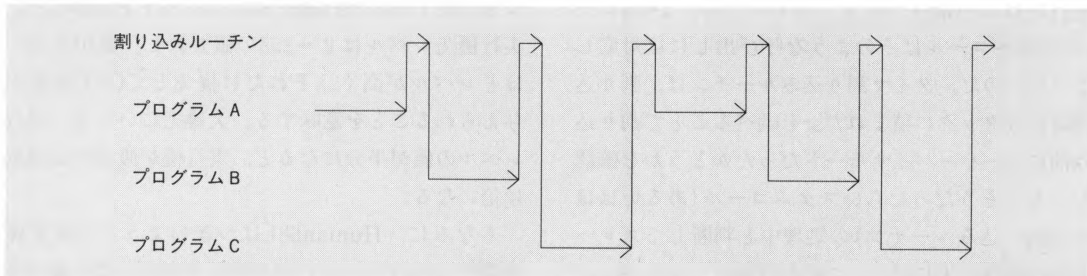
一般に，タスク[1]の切り替えは一定時間ごとに割り込みをかけるタイマを利用して行われる。図1にその様子を示した。横軸に時間，縦軸に実行中のタスク(概念上はプログラムカウンタ)をとってある。左からみてもらうと，最初はプログラムAが走っている。ここにタイマ割り込みがかかり，制御は割り込みルーチンに移る。タスク切り替えはこのタイマ割り込みルーチンが行う。通常の場合，割り込みルーチンでは必要な処理がすんだら割り込み発生時に実行中だったプログラムに戻るわけだが，故意に，戻りアドレスを操作してべつのプログラム(図ではプログラムB)に制御を移すのだ。この際，本来戻るはずだったプログラムA内の位置や割り込みがかかった時点でのレジスタ内容はどこかに保存しておき，つぎにプログラムAの実行を再開するときに元に戻す。こうして割り込みルーチン内でレジスタ内容をこっそり，かつ，ごっそり，とっかえひっかえすること＝タスク切り替え，となる。Human68kのバックグラウンド処理機能も，基本的にはこのような方法で複数プログラムを並列実行する。

さて，通りのよさから便宜上タスクという言葉を使ってきたが，Human68kのバックグラウンド処理機能では“スレッド(thread)”が並列処理の管理単位になる。スレッドとは，何かひとつながりのものといった意味で，いまの場合，逐次的に処理されるプログラムの命令列，あるいは，その処理の流れを指す。各スレッドはスレッドIDと呼ばれる番号とスレッド名で識別される[2]。起動時にはメイン(スレッドIDが0，スレッド名“Human68k system”)のスレッドがひとつ用意され，フォアグラウンドで動くプログラム，つまり，COMMAND.Xやその子プロセスとして走る各プログラムはこのメインのスレッドに属する[3]。このメインスレッドだけの状態からDOSコールopen_prによりタスクを登録する

1) タスク(task)とは，ひとまとまりの処理単位を指す言葉だ。その厳密な意味や単位の大小はOSごとに定義されうる。本稿では漠然とした“処理単位”の意味でのみ，この言葉を使う。

2) PROCESS.Xに/Aまたは/Bオプションを与えて実行すると，全スレッドのスレッドID，スレッド名を知ることができる。

3) Human68kでは子プロセスを生成しても，制御が親から子へ，子から親へ，と移るだけで処理の流れは一本道だ。

図1　タイマ割り込みによるタスク切り替え

と新たなスレッドが生まれて，メインのスレッドとCPU時間を分け合いながら実行されることになる。

バックグラウンドタスクはかならず常駐プログラムとして作成し，open_pr後，DOSコールkeepprで常駐終了する。このとき，必要なら同じメモリブロック上に複数のタスクを置き，個別にopen_prで登録してからまとめて常駐することも可能だ。その場合，各タスクは同一プロセスとして環境を共有することになる。

登録したバックグラウンドタスクを破棄するときには，DOSコールkill_prを使う。kill_prはそれを呼び出したタスク自身の属するスレッドを破棄し，常駐して占有していたメモリも解放する。複数タスクからなるバックグラウンドプロセスの場合，あるタスクが自殺すると，同一プロセスとして環境を共有する全タスクも道連れになる。ここで，kill_prはあくまで“自殺”するDOSコールだ。自分以外のスレッドを破棄する直接的な方法は用意されていない。その場合は，スレッド間通信により破棄要求を送りつけて自殺させるという方法をとる。スレッド間通信についてはまたあとで触れよう。

スレッド切り替えの実際

スレッド切り替え用のタイマとしてはMFPのタイマDが使用される。タイマDの割り込みを設定するIOCSコールTIMERDSTは，PROCESSを設定した時点で無効化され，以後，ユーザーからは利用不能になる。なお，本来タイマDの割り込み間隔は細かく設定できるのだが，最長に設定してもそれをそのままスレッド切り替え間隔とするには短すぎるため，実際には割り込み間隔を1ms固定とし，割り込みルーチン側で割り込み回数を数えて一定回数ごとにスレッドを切り替えるようになっている。

割り込みによるスレッド切り替えは，ユーザーモード時にかぎって行われる。これはDOSコールやIOCSコールなどのシステムコールがリエントラント(re-entrant：再入可能)になっていないためだ。仮にシステムコールの処理中でもスレッド切り替えを許すとなると，システムコールの処理が終わりに達しないうちにほかのスレッドから同じシステムコールが発行される可能性が出てくるわけだが，DOSコールやIOCSコールはそのような呼び出しには対応していないのだ。タイマ割り込みルーチンは，割り込み時にスタックに積まれたsrを調べることで割り込み前にスーパーバイザモードだったかどうかを確認し，もしそうだったらシステムコール(あるいはほかの割り込みルーチン)の処理中と判断してスレッドを切り替えずに戻る。この副作用で，スーパーバイザモードで走るアプリケーションプログラムの実行中もスレッド切り替えは停止する。そこで，スーパーバイザモードで走るプログラムは，自発的に実行権を放棄してほかのスレッドに実行権を渡すDOSコールchange_prを定期的に呼び出すことが推奨されている。

もっとも，タイマ割り込みルーチンはスレッドを切り替えそこねたときには内部のフラグを立てて戻るようになっており，Human68k本体はDOSコールの処理が終わるたびにそのフラグを調べ，必要なら(割り込みによらずに)スレッドを切り替えてくれる。このため，スーパーバイザモードで走るプログラムであっても，適当な間隔でDOSコールを発行していれば，スレッド切り替えを完全に止めてしまうことはない。むしろ，単に実行権を占有してしまわないようにするだけの目的であれば，change_prよりもダミーのDOSコールを発行したほうがよいかもしれない。change_prを使うと，まだそのスレッドが使っていられる時間が残っていても無条件にスレッドが切り替わってしまうが，そのほかのDOSコールなら時間切れになっていたときにのみスレッドが切り替わる。このとき利用するDOSコールとしては，害がなく，ごく短期間で処理が終わるもの，たとえば，curdrvなんかが適しているだろう。

このほか，スレッド切り替えはDOSコール中でキー入力待ちになったときにも行われる。キー入力など，ユーザーの応答を待っているあいだは最もCPUが遊んでいる期間であり，キー入力があるまでほかのスレッドに実行権を渡すのはごく自然だ。そもそも(シングルユーザー環境での)マルチタスクの存在意義はこのような無駄な時間を有効利用することにある。ここで，IOCSコールB_KEYINPにはこのような細工が施されていないことに注意したい。キーの先行入力がないときにB_KEYINPを呼び出すと，キー入力があるまでスレッド切り替えが停止してしまう。そうしないためには，B_KEYSNSでキーバッファにデータがあるのを確かめてからB_KEYINPを呼び出すようにしなければならない。

さて，スレッド切り替えは単純にスレッドID順に行われるわけではない。各スレッドには実行優先レベル[4]が与えられており，Human68kはこのレベルを考慮して，実行権を渡すスレッドを決める[5]。実行優先レベルは2～255の値で表し，値が小さいほどレベルが高く，それだけ優先してCPU時間が与えられることを意味する。大雑把にいって，実行レベルの値が半分になると，実行権が渡される回数は倍になる。

ちなみに，Human68kはつぎのような方法で優先順位つきのスレッド切り替えを実現している。ま

4) PROCESS.Xの出力するスレッド情報では“モード”の欄に表示される。

5) 実行権をどのプログラムに渡すかを決める処理をスケジューリングという。

ず，各スレッドごとに1バイトのカウンタを用意し，実行優先レベル－1の値で初期化しておく。スレッド切り替え時には，カウンタの値が最小のスレッドを探す。この検索はいままで実行していたスレッドのつぎのスレッドからスレッドID順に行われる。カウンタが最小のスレッドが複数あった場合は先にみつかったほうが選ばれる。こうしてみつけたスレッドに実行権を渡すのだが，このとき，そのスレッドのカウンタの残りをほかのスレッドのカウンタから引き，同時に実行権が渡されたスレッドのカウンタをリセットする。以下，この繰り返しだ。参考までに実行優先レベル3，4，6の3スレッドがある場合のカウンタの変化を表1に示しておこう(表中，矢印はカウンタのリセットを表す)。

ところで，実行優先レベルはopen_prでタスクを登録するときに(メインスレッドの場合はCONFIG.SYSのPROCESS行で)指定できるのみで，あとから変更する方法が用意されていない。そこでリスト1だ。たぶん，今月唯一の実用プログラムだろう。引数としてスレッドIDと実行優先レベルを指定すると，対応するHuman68k内部のスレッド管理情報を直接書き換えて，実行優先レベルを変更する。アセンブル時にはリスト2もカレントディレクトリに置いておく。リスト2はバックグラウンド処理関係の構造体/定数の定義ファイルであり，あとのプログラムでも共通に使う。

リスト1はHuman68kのワークを書き換えてしまうという点で悪いプログラムだが，書き換える位置を算出する手順は正当といえるものだ。基本線では公開されている情報しか使っておらず，Human68k内部のワークを絶対アドレスで参照しているわけではない。人によっては，どうせ悪さをするならHuman68kのワークを覗いてスレッド情報のアドレスを得てしまえばよいと考えるかもしれないが，ほかに手があるなら少々まどろっこしくてもそっち

表1　優先順位つきのスレッド切り替え

スレッドA	スレッドB	スレッドC	実行されるスレッド
2	3	5	初期状態
0→2	1	3	スレッドA
1	0→3	2	スレッドB
0→2	2	1	スレッドA
1	1	0→5	スレッドC
0→2	0	4	スレッドA
2	0→3	4	スレッドB
0→2	1	2	スレッドA
1	0→3	1	スレッドB
0	2	0→5	スレッドC
0→2	2	5	スレッドA
0	0→3	3	スレッドB
0→2	3	3	スレッドA
0→2	1	1	スレッドA
1	0→3	0	スレッドB
1	3	0→5	スレッドC

リスト1　RENICE.X

```
 1: *           スレッドの実行優先レベルを変更する
 2:
 3:             .include        doscall.mac
 4:             .include        iocscall.mac
 5: *           .include        fefunc.h
 6:             .include        bg.h
 7:             .include        const.h
 8: *
 9: FPACK       macro   callno
10:             .dc.w   callno
11:             .endm
12: *
13: __STOL      equ     $fe10
14: *
15:             .text
16:             .even
17: *
18: ent:
19:             lea.l   inisp(pc),a7
20:
21:             tst.b   (a2)+
22:             beq     usage
23:
24:             movea.l a2,a0
25:             FPACK   __STOL
26:             bcs     usage
27:             move.w  d0,d2                   *d2.w = スレッドID
28:
29:             FPACK   __STOL
30:             bcs     usage
31:             move.b  d0,d1                   *d1.b = レベル
32:             subq.b  #2,d0
33:             bcc     do
34:             moveq.l #2,d1
35:
36: do:         lea.l   thinfo(pc),a1           *指定IDを持つ
37:             pea.l   (a1)                    *  スレッドの管理情報を得る
38:             move.w  d2,-(sp)                *
39:             DOS     _GET_PR                 *
40: *           addq.l  #6,sp                   *
41:             tst.l   d0                      *
42:             bmi     nfound                  *
43:
44:             move.l  (a1),d3                 *d3 = 次のスレッド管理情報
45:
46:             move.l  d3,d0                   *スレッド管理情報の
47:             moveq.l #thNMAX-1,d4            *  本体を探す
48: sealp:      movea.l d0,a1                   *
49:             IOCS    _B_LPEEK                *
50:             cmp.l   d3,d0                   *
51:             dbeq    d4,sealp                *
52:             bne     nfound                  *
53:
54:                                             *a1 = スレッド管理情報の本体+4
55:             addq.l  #thLEVEL-4,a1           *実行優先レベルを書き換える
56:             subq.b  #1,d1                   *
57:             IOCS    _B_BPOKE                *
58:
59:             DOS     _EXIT
60: *
61: usage:      lea.l   usgmes(pc),a0
62:             bra     error
63: nfound:     lea.l   errmes(pc),a0
64:             *
65: error:      move.w  #STDERR,-(sp)
66:             pea.l   (a0)
67:             DOS     _FPUTS
68: *           addq.l  #6,sp
69:
70:             move.w  #1,-(sp)
71:             DOS     _EXIT2
72: *
73: usgmes:     .dc.b   '機  能: スレッドの実行優先レベルを'
74:             .dc.b   '変更する',CR,LF
75:             .dc.b   '使用法: RENICE スレッドID(0~31)'
76:             .dc.b   ' レベル(2~255)',CR,LF,0
77: errmes:     .dc.b   '指定のスレッドが見つかりません',CR,LF,0
78: *
79:             .bss
80:             .even
81: *
82: thinfo:     .ds.b   SIZEofTHREADINFO
83: *
84:             .stack
85:             .even
86: *
87:             .ds.l   256
88: inisp:
89:
90:             .end    ent
```

を使うのが僕の美意識だ[6]。

で，リスト1ではget_prで指定IDのスレッド情報のコピーを得たら，その先頭にある“つぎのスレッド情報を指すポインタ”をたどって，コピーと同じものがHuman68k内部のどこにあるかを探している。スレッド情報はHuman68k内部でもget_prの返す形式(リスト2：11～33行)のまま保持されており，また，最後のスレッド情報と先頭のスレッド情報が輪になっている[7]ので，この方法はうまく働く。スレッドID 0のスレッド情報を得て，必要な回数だけポインタをたどる方法もあるが，その場合，引数で指定されたスレッドIDが有効かどうかをべつに調べなければならない。リスト1では指定のスレッドIDが有効かどうかはget_prを呼び出したときにわかる。

スレッドの状態とスレッド間通信

スレッドにはアクティブ状態とスリープ状態がある。字面から読み取れると思うが，ふつうに動いている状態をアクティブ，一時的に停止して待機している状態をスリープと表現する。スリープ状態のスレッドには実行権が回ってこない。

バックグラウンドタスクはかならずしもひっきりなしに走っていなくてもよい場合が多い。たとえば，時計プログラムの場合，基本的には1秒に1回画面を書き換えればよいわけだ。このように，1回まとまった処理をしたらしばらくすることがないとき，バックグラウンドタスクは自発的に一定期間スリープしてCPU時間の浪費を防ぐのが礼儀とされる[8]。

スリープするにはDOSコールsleep_prを使う。スリープする時間は引数で1ms単位で指定できる。特に0を指定した場合は無限と解釈され，外部からスレッド間通信により起こされるまでスリープし続ける。また，open_prで生成された直後のスレッドは無条件にスリープ状態になる。このときの待ち時間はopen_prの最後の(最初にスタックに積む)引数で指定する。

自分以外のスレッドをスリープさせる方法は2通りある。ひとつは，スレッド間通信でスリープするよう要求する方法だ。この場合，通信を受け取ったスレッドが自発的にスリープすることになるので，そのプログラムがスリープ要求コマンドをサポートしていなければ意味がない[9]。強制的にスリープさせるには，DOSコールsuspend_prを使う[10]。suspend_prはスレッド間通信を使わずに，強制的に指定スレッドをスリープ状態にする[11]。suspend_prで止められたスレッドは，やはりスレッド間通信で起こされるまでスリープし続ける。

たびたび登場したスレッド間通信はDOSコールsend_prを使って行う。send_prでは任意のスレッドに対して，16ビットのコマンドと，もし必要なら不定長の付随データを送ることができる。送り先スレッドはスリープ状態であってもよい。通信を送られたスレッドは自動的にアクティブ状態になる。このとき，受け手側ではsleep_prからの戻り値により，通信によって起こされたのか，待ち時間が過ぎたのかを区別できる。なお，すでに通信が入っている状態でスリープしようとしてもsleep_prからはすぐに戻り，スリープできないようになっている。

6) だから，僕なら日本語入力フロントエンドプロセッサのファンクションコールを利用してワープロを作るときにファンクションコールの仕様が少々気に入らないからとしても，内部ワークを書き換えてつじつまを合わせることを考える前に相当品を自前で用意するだろう。少なくとも，ローマ字かな変換ならIOCSのサポートもあることだし……って何の話だよ。

7) 結局は未公開情報だったりする。

8) 実際，TIMER.Xは1秒に1回しか起きないので，ほとんどフォアグラウンドタスクに負担をかけない。だが，本当はもう少し細かく起きないと，秒の変わり目がうまく検出できないはずだ……と思ってX-BASICで時刻を表示し続けるプログラムを書いてTIMER.Xが表示する時計と比べてみたら，しっかりと，最大1秒近く画面の更新が遅れていた。

リスト2 BG.H

```
 1: *           バックグラウンドプロセス用定数/構造体定義
 2:
 3: thNMAX              equ     32          *最大スレッド数
 4: *
 5: thUSER              equ     $0000       *ユーザーモードで走る
 6: thSUPER             equ     $2000       *スーパーバイザモードで走る
 7:
 8: *
 9: *           スレッド管理情報
10: *
11:                     .offset 0
12: *
13: thNEXT:             .ds.l   1           *つぎのスレッド
14: thWAITFLAG:         .ds.b   1           *スリープ中かどうかのフラグ
15:                                         *   00h ... 起きている
16:                                         *   FEh ... 強制スリープ中
17:                                         *   FFh ... スリープ中
18: thCOUNT:            .ds.b   1           *実行優先順位づけ用カウンタ
19: thLEVEL:            .ds.b   1           *実行優先レベル-1
20: thDOSCOMMAND:       .ds.b   1           *実行中だったDOSコール番号
21: thPSP:              .ds.l   1           *psp
22: thUSP:              .ds.l   1           *usp
23: thDREG:             .ds.l   8           *d0~d7
24: thAREG:             .ds.l   7           *a0~a6
25: thSR:               .ds.w   1           *sr
26: thPC:               .ds.l   1           *pc
27: thSSP:              .ds.l   1           *ssp
28: thINDOSF:           .ds.w   1           * (DOSコールネスティングレベル)
29: thINDOSP:           .ds.l   1           * (DOSコール処理中のssp)
30: thBUFF:             .ds.l   1           *スレッド間通信バッファ
31: thNAME:             .ds.b   16          *スレッド名
32: thWAITTIME:         .ds.l   1           *スリープ時間残り
33: SIZEofTHREADINFO:
34:                     .text
35: *
36: thNAMED             equ     -1          *スレッド名で指定する
37: thMYSELF            equ     -2          *自分自身
38:
39: *
40: *           スレッド間通信バッファ
41: *
42:                     .offset 0           *スレッド間通信バッファ
43: *
44: thCOMMDATALEN:      .ds.l   1           *バッファバイト数
45:                                         * (通信時はデータバイト数)
46: thCOMMDATA:         .ds.l   1           *データ格納領域先頭アドレス
47: thCOMMCOMMAND:      .ds.w   1           *コマンド
48: thCOMMID:           .ds.w   1           *送り手ID
49: SIZEofCOMMBUFF:
50:                     .text
51: *
52: thEMPTY             equ     -1          *スレッド間通信バッファにデータがない
53:
54: *
55: *           スレッド間通信コマンド
56: *
57: thKILL              equ     $fff9       *破棄要求
58: thWAKEUP            equ     $fffb       *アクティブにする
59: thSLEEP             equ     $fffc       *スリープ要求
60: thISBUSY            equ     $ffff       *スレッド間通信可能かどうか調べる
61:
62: *
63: *           エラーコード
64: *
65: thCONFLICT          equ     -27         *同名のスレッドが存在する
66: thBUSY              equ     -28         *スレッド間通信が受けつけられない
67: thNOMORE            equ     -29         *これ以上スレッドを登録できない
```

通信内容は各スレッドごとに用意された通信バッファに書き込まれる。通信バッファはリスト2でいうと42～49行のような構造をしている。通信を受け取るためには，事前にデータ格納領域の先頭アドレスとその大きさを設定し，また，送り手のスレッドID格納フィールドに－1を入れておく必要がある。スリープしないプログラムの場合は，このID格納フィールドが－1かどうかで通信があったかどうかを判断する。ここで，通信が入るとデータ領域先頭アドレス以外のフィールドは上書きされることになるので，再び通信を受けつけるようにするには，通信の処理が済み次第，バッファをリセットしないとならないことに注意しよう。

コマンドコードは基本的には各アプリケーションが独自に意味を与えて使用することができる。ただし，$FFxx_H$のコマンドコードはシステムによって予約されており，とくにその一部にはリスト2の57～60行のような意味が定義されている。破棄要求($FFF9_H$)とスリープ要求($FFFC_H$)についてはすでに触れた。$FFFB_H$は特別なコマンドであり，通信先のスレッドをアクティブにするだけで，スレッド間通信バッファは変化させない(＝実際には通信しない)。これは，スレッド間通信の処理途中でsuspend_prで止められたスレッドを起こすときに，通信バッファ内容を変えてしまわないための仕様だろう

通信可能かどうかの検査コマンド($FFFF_H$)はとにかく通信してみて，受け付けられたかどうかを調べるコマンドだ。送り先の通信バッファにまだ未処理のデータがあるとsend_prはエラーコード－28を返すので，通信可能かどうかがわかる。ただ，このコマンドを実際に使うことはあまりないだろう。アクティブにするコマンド$FFFB_H$同様の特別扱いにはなっておらず実際にコマンドが送られるため，続けてべつのコマンドを送るには送り先が検査コマンドを処理してスレッド間通信バッファをリセットするのを待たなければならない。これなら最初から送りたいコマンドを直接送って，エラーの有無を調べたほうが利口だ。

スレッド間通信の利用例として，リスト3に任意のスレッドを破棄するプログラムを示す。単に破棄要求コマンドを送っているだけなので，send_prの使用例としての意味しかないが，29行のコマンドコードを変えれば，スリープ要求を送るプログラム，アクティブにするプログラムにもすぐ化ける。また，スレッド間通信からは離れるが，27～33行を，

```
move.w   d0,-(sp)
DOS      _SUSPEND
addq.l   #2,sp
```

にすれば，強制スリープさせるプログラムにもなる。ひととおり用意しておけば，何かの役には立つだろう。

ただ，リスト3には手抜きがいっぱいある。第1に，本来send_prでは引数で送り主のスレッドIDを指定して自分の身分を明らかにすることになっているのだが[12]，リスト3では自分がメインスレッドであることを仮定して(その仮定は通常成り立つが)，自分のスレッドIDを取得するのを怠っている。第2に，相手が通信を受け取ったかどうかを確認していない。第3に，相手がちゃんと自己破棄したかどうかも確認していない。第4に，そもそも破棄要求はあまりうかつに送ってよいものではない。プログラムによっては，破棄に先立ってべつのプログラム(通常，そのプログラムの非常駐部)がある程度のあと始末をすることを前提にしている場合がある。いちおう，TIMER.XとPRINT.Xでは問題ないようだが，ほかのプログラムでは誤動作することもある

9) 純正のバックグラウンド処理プログラムTIMER.XとPRINT.Xはともにスリープ要求コマンドをサポートしていない。

10) DOSCALL.MAC中でのコール名は_SUSPENDになっている。

11) 正確にいうと，内部での扱いはsleep_prによる自発的なスリープ状態とは区別されている。

12) Human68kはsend_prを呼び出したスレッドのIDを知っているのだから，わざわざ引数で指定する意味はないように思うのだが，そういう仕様だ。

リスト3 KILL.S

```
 1: *         スレッドを破棄する
 2:
 3:           .include       doscall.mac
 4: *         .include       fefunc.h
 5:           .include       bg.h
 6:           .include       const.h
 7: *
 8: FPACK     macro   callno
 9:           .dc.w   callno
10:           .endm
11: *
12: __STOL    equ     $fe10
13: *
14:           .text
15:           .even
16: *
17: ent:
18:           lea.l   inisp(pc),a7
19:
20:           tst.b   (a2)+
21:           beq     usage
22:
23:           movea.l a2,a0
24:           FPACK   __STOL
25:           bcs     usage
26:
27:           clr.l   -(sp)
28:           clr.l   -(sp)
29:           move.w  #thKILL,-(sp)
30:           move.w  d0,-(sp)
31:           clr.w   -(sp)
32:           DOS     _SEND_PR
33: *         lea.l   11(sp),sp
34:
35:           DOS     _EXIT
36: *
37: usage:    move.w  #STDERR,-(sp)
38:           pea.l   usgmes(pc)
39:           DOS     _FPUTS
40: *         addq.l  #6,sp
41:
42:           move.w  #1,-(sp)
43:           DOS     _EXIT2
44: *
45: usgmes:   .dc.b   '機　能: スレッドを破棄する',CR,LF
46:           .dc.b   '使用法: KILL スレッドID',CR,LF,0
47: *
48:           .stack
49:           .even
50: *
51:           .ds.l   256
52: inisp:
53:
54:           .end    ent
```

だろう。

バックグラウンド処理プログラム

そろそろ実際にバックグラウンド処理を行うプログラムの例を示すとしよう。リスト4は単独でアセンブル/リンクして実行すると常駐し，ファンクションキー行の左端にA～Zの文字を順に表示し続ける。/Rオプションによりスレッドを破棄して常駐解除する。まったく実用性のないプログラムではあるが，バックグラウンド処理プログラムとしてはかなり真面目に作ってあるので，たぶん，このままスケルトンとして利用できると思う。

非常駐部からみていこう。常駐プログラムの鉄則どおり，非常駐部は常駐部のうしろ，97行から始まる。100～102行ではHuman68kのバージョンを確認している。先月触れそこねたのだが，Human68k ver 2.0の機能を利用する場合はバージョン番号が2.00以降であることを確認しなければならない。なお，『プログラマーズマニュアル』のサンプルプログラムではこれに加えてバージョン2.50未満であることを確認しているが，理由が示されていないし，TIMER.Xはそうなっていないようなのでここではコメントで殺してある。好みに応じて復活させてもらいたい。

106～123行で/Rオプションの指定の有無を確認し，指定がなければ185行の常駐処理に飛ぶ。通常の常駐プログラムよりも常駐するための処理が簡単になっていることに注目してほしい。常駐プログラムでは多重常駐を防ぐために，すでに自分と同じプログラムが常駐しているかどうか調べる必要があるわけだが，バックグラウンド処理プログラムの場合，open_prですでに存在するスレッドと同名のスレッド名でタスクを登録しようとするとエラーが返るので，特に気を払わなくても多重常駐が防げる[13]。あとは常駐メッセージを出して，常駐終了するだけだ。ただ，リスト4ではここで一瞬危険区域を通過する。もし，open_prから戻って常駐終了するまでのあいだに何らかの理由で実行が中断されると，スレッドが生成されたのに常駐しないで終了することになり，間違いなく暴走するだろう。以前，常駐プログラムを作ったときに，そのようなことがないよう中断時の戻りアドレスを設定して再試行する方法を示したが，このプログラムではどうやっても危険区域が残ってしまうようなので，開き直って無視している。あとはタイミング悪くキーボードの上に何かが降ってきてBREAKキーやCOPYキーなどを押してしまわないことを祈ろう。

危険区域といえば，リスト4では常駐部のスタックを非常駐部と重ねて確保している(88～89行)ため，スレッドを生成してから常駐終了するまでのあいだは常駐部のスタックの中を走る格好になる。いちおう注意して作ってあるので問題は起きないが，リスト4を流用する場合はスタックと常駐部の大きさや位置関係には気をつけてもらいたい。

もうひとつ，スタックで思い出した。バックグラウンド処理プログラムでは少なくともスーパーバイザスタックとして6Kバイトを確保することになっている。実際にはこれにユーザースタックが加わる。リスト4のように常駐部のスタックを非常駐部と重ねて確保する場合，6Kバイトもあれば非常駐部はスタックに使う分を含めて十分収まる。しかし，収めてしまってはならない。収まってしまう場合は，非常駐部用のスタックを大きめにとり(リスト4もそうなっている)，故意に常駐サイズよりも広いメモリを確保しておく。プログラムが起動時に与えられるのはスタックセクションの末尾までで，そのうしろにメモリがまだあるという保証はない[14]。それ以上の大きさのメモリを確保する形で常駐できるかどうかはわからないのだ。リスト4のようにequで常駐部の末尾アドレスを表す場合は気がつかないうちにはみ出してしまっていることがあるので，十分気をつけるようにしたい。

128行以下の常駐解除処理では，常駐部に破棄要求を送り，常駐部が自殺するのを待つ，という処理をしている。一般の常駐プログラムのように常駐部がどこにあるのか探してからメモリを解放するといった手順を踏む必要はない。まず，128～139行で破棄要求の送り先スレッドのスレッドIDを得る。通常，get_prはスレッドIDで指定したスレッドの管理情報を返すわけだが，管理情報を受け取るメモリ領域のスレッド名が格納される部分にスレッド名を書き込んでおいてからスレッドIDに－2を指定すると，

13) 逆に故意に多重常駐する場合，スレッド名を変えて何度かリトライする必要も出てくるだろう。

14) あるかどうかを確認することはできる。

リスト4 BGTEST.S

```
 1: *         バックグラウンドプロセスのテストプログラム
 2:
 3:          .include        doscall.mac
 4:          .include        iocscall.mac
 5:          .include        bg.h
 6:          .include        const.h
 7: *
 8:          .offset 0                    *常駐部ワークの構造
 9: *
10: combuf:  .ds.b   SIZEofCOMMBUFF              *通信バッファ
11: a6sav:   .ds.l   1                           *a6待避用
12: spsav:   .ds.l   1                           *sp待避用
13: SIZEofWORK:
14: *
15:          .text
16:          .even
17: *
18: *        常駐部
19: *
20: keepst:
21: bgent:   lea.l   work(pc),a6                 *ワークアクセス用
22:          movem.l a6/sp,a6sav(a6)             *中断時に備えて
23:                                              *  a6,spを待避
24:
25:          pea.l   break(pc)                   *中断時処理アドレスを
26:          move.w  #_CTRLVC,-(sp)              *  設定
```

get_prがスレッド名に対応するスレッドを探してくれるようになっている。ここでエラーが返るようなら，まだ常駐していなかったことになる。続いて，141～144行で自分のスレッドIDを得る。スレッドIDに－1を指定すると自分のスレッド管理情報とスレッドIDが返る。

146行以降は，先ほど手を抜いたスレッド間通信のより正しい姿を示している。146～151行ではとりあえずアクティブにするコマンドを送って，指定のスレッドを起こす。これは，リスト3を改造したプログラムなどを使って，通信途中にsuspend_prにより強制スリープ状態にされている可能性を考慮したものだ。それから，159行で実際に破棄要求を送る。このとき，通信バッファにまだ未処理の通信データが残っていることを表すエラーコードが返ってきたら，受けつけられるまで要求を送り続ける。ここで，単にsend_prをループで括っただけではあまり意味がないことに注意してほしい。1回送信するたびに，いったんchange_prで実行権を放棄して，常駐部にスレッド間通信バッファをリセットする時間を与えなければならない。単にループしていてもいつかはスレッドが切り替わるが，change_prを使ったほうが少なくともプログラムはわかりやすくなる。

うまく通信できるか，あるいは，send_prが予期せぬエラーコードを返した場合はループを抜けて164行にくる。エラーが返った場合はあきらめてエラー終了する。すでに送り先のスレッドが存在することは確認しているし，アクティブにもしてあるから，変なエラーコードが返ってきたら，それはかなりの異常事態だ。ただ，稀なケースながら，151行でアクティブにしたあとにそのスレッドが破棄されてしまった，という可能性もゼロではない。たとえば，すでに破棄要求が送られていて，それを処理する前にsuspend_prで止められていたのであれば，151行でアクティブにしたことで動き出してすぐ自殺する。重箱の隅をつつくような話だが，複数プログラムが並行動作する環境では，ほんの少し前に得た情報がいつまでも正しいとはかぎらないということは頭に入れておいたほうがよいだろう。

無事，通信が送れたら，今度はそれを常駐部が受け取って，要求どおりに自己破棄を行うことを確認する。get_prでスレッド情報を取得して，うまく取得できるあいだはそのスレッドがまだ生きている。逆にget_prがエラーを返したらそのスレッドが死んだと判断できる。ここでも，自殺する時間を与えるために合間にchange_prを呼び出して実行権を回してやるのを忘れない。なお，『プログラマーズマニュアル』にはchange_prとget_prを1回ずつ呼び出せば，破棄がうまくできたかできなかったを判断でき

```
 27:          DOS      _INTVCS                       *
 28:          move.w   #_ERRJVC,(sp)                 *
 29:          DOS      _INTVCS                       *
 30:          addq.l   #6,sp                         *
 31:
 32: loop:    cmpi.w   #thEMPTY,combuf+thCOMMID(a6)
 33:          beq      main                          *スレッド間通信バッファに
 34:                                                 *  データはあるか?
 35:
 36:          move.w   combuf+thCOMMCOMMAND(a6),d0
 37:                                                 *d0 = コマンド
 38:          move.w   #thEMPTY,combuf+thCOMMID(a6)
 39:                                                 *通信を許可する
 40:
 41:          cmpi.w   #thKILL,d0                    *破棄要求?
 42:          bne      nkill
 43:
 44:          DOS      _KILL_PR                      *自身を破棄する
 45:          bra      loop
 46:
 47: nkill:   cmpi.w   #thSLEEP,d0                   *スリープ要求?
 48:          bne      main
 49:
 50:          clr.l    -(sp)                         *永久スリープする
 51:          DOS      _SLEEP_PR                     *
 52:          addq.l   #4,sp                         *
 53:          bra      loop
 54:
 55: main:    moveq.l  #3,d1                         *画面左下に
 56:          moveq.l  #0,d2                         *  'A'～'Z'を
 57:          moveq.l  #31,d3                        *  順に表示する
 58:          moveq.l  #1-1,d4                       *
 59:          lea.l    char(pc),a1                   *
 60:          addq.b   #1,(a1)                       *
 61:          cmpi.b   #'Z'+1,(a1)                   *
 62:          bcs      put                           *
 63:          move.b   #'A',(a1)                     *
 64: put:     IOCS     _B_PUTMES                     *
 65:
 66: wait:    pea.l    100.w                         *0.1秒ほど
 67:          DOS      _SLEEP_PR                     *  スリープする
 68:          addq.l   #4,sp                         *
 69:          bra      loop
 70:
 71: char:    .dc.b    '@',0
 72:
 73: *
 74: *        中断時処理
 75: *
 76: break:   movem.l  work+a6sav(pc),a6/sp         *レジスタを復帰する
 77:          bra      loop
 78:
 79: *
 80: *        常駐部ワーク
 81: *
 82: work:    .dc.l    0                             *スレッド間通信バッファ
 83:          .dc.l    0                             *
 84:          .dc.w    0                             *
 85:          .dc.w    thEMPTY                       *
 86:          .ds.b    SIZEofWORK-SIZEofCOMMBUFF
 87: *
 88: iniusp   equ      *+1024              *ユーザースタック末尾
 89: inissp   equ      iniusp+1024*6       *スーパーバイザスタック末尾
 90: keeped   equ      inissp              *常駐部末尾
 91: KEEPSIZ  equ      keeped-keepst       *常駐部バイト数
 92:
 93: ***************************************************************
 94: *
 95: *        非常駐部
 96: *
 97: ent:
 98:          lea.l    inisp(pc),sp
 99:
100:          DOS      _VERNUM                       *Human68kの
101:          cmpi.w   #$0200,d0                     *  バージョンを確認
102:          bcs      vererr                        *
103: *        cmpi.w   #$0232,d0                     *
104: *        bcc      vererr                        *
105:
106:          tst.b    (a2)+                         *引数がなければ
107:          beq      keep                          *  常駐処理へ
108: skipsp:  move.b   (a2)+,d0                      *
109:          beq      keep                          *
110:          cmpi.b   #SPACE,d0                     *
111:          beq      skipsp                        *
112:          cmpi.b   #TAB,d0                       *
113:          beq      skipsp                        *
114:
115:          cmpi.b   #'-',d0                       *オプションの検査
116:          beq      chkopt                        *
117:          cmpi.b   #'/',d0                       *
118:          bne      usage                         *
119:                                                 *
120: chkopt:  moveq.l  #$20,d0                       *
121:          or.b     (a2)+,d0                      *
122:          cmpi.b   #'r',d0                       *-rオプション?
123:          bne      usage                         *
124:
125: *
```

```
126: *           常駐解除処理
127: *
128: remove: lea.l   myname(pc),a0             *自分と同じ名前の
129:         lea.l   thinfo+thNAME(pc),a1      *  スレッドがあるか?
130: cpylp:  move.b  (a0)+,(a1)+               *
131:         bne     cpylp                     *
132:                                           *
133:         pea.l   thinfo(pc)                *
134:         move.w  #thNAMED,-(sp)            *
135:         DOS     _GET_PR                   *
136: *       addq.l  #6,sp                     *
137:         tst.l   d0                        *
138:         bmi     rmerr1                    *なかった
139:         move.w  d0,d1                     *d1 = スレッドID
140:
141:         pea.l   thinfo(pc)                *自分のスレッドIDを得る
142:         move.w  #thMYSELF,-(sp)           *
143:         DOS     _GET_PR                   *
144: *       addq.l  #6,sp                     *d0 = 自分のスレッドID
145:
146:         clr.l   -(sp)                     *とりあえず起こす
147:         clr.l   -(sp)                     *
148:         move.w  #thWAKEUP,-(sp)           *
149:         move.w  d1,-(sp)                  *
150:         move.w  d0,-(sp)                  *
151:         DOS     _SEND_PR                  *
152:
153:         move.w  #thKILL,4(sp)             *受け付けられるまで
154:         moveq.l #thBUSY,d2                *  破棄要求を送る
155:         bra     send                      *
156:                                           *
157: sendlp: DOS     _CHANGE_PR                *
158:                                           *
159: send:   DOS     _SEND_PR                  *
160:         cmp.l   d2,d0                     *
161:         beq     sendlp                    *
162: *       lea.l   14(sp),sp                 *
163:
164:         tst.l   d0                        *異常発生
165:         bmi     rmerr2                    *
166:
167:         pea.l   thinfo(pc)                *死んだの確認する
168:         move.w  d1,-(sp)                  *
169: waitlp: DOS     _CHANGE_PR                *
170:                                           *
171:         DOS     _GET_PR                   *
172:         tst.l   d0                        *
173:         bpl     waitlp                    *
174:         addq.l  #6,sp                     *
175:
176:         pea.l   remmes(pc)                *常駐解除メッセージを出力
177:         DOS     _PRINT                    *
178: *       addq.l  #4,sp                     *
179:
180:         DOS     _EXIT
181:
182: *
183: *       常駐処理
184: *
185: keep:                          *バックグラウンドタスクを登録する
186:         pea.l   1.w                       *スリープ時間1ms
187:         pea.l   work+combuf(pc)           *スレッド間通信バッファ
188:         pea.l   bgent(pc)                 *初期pc (実行開始アドレス)
189:         move.w  #thUSER,-(sp)             *初期sr (ユーザーモード)
190:         pea.l   inissp(pc)                *初期ssp
191:         pea.l   iniusp(pc)                *初期usp
192:         move.w  #2,-(sp)                  *実行優先レベル 最高
193:         pea.l   myname(pc)                *スレッド名
194:         DOS     _OPEN_PR                  *
195: *       lea.l   28(sp),sp                 *
196:         tst.l   d0                        *
197:         bmi     kperr                     *登録失敗
198:
199:         pea.l   keepms(pc)                *常駐メッセージを出力
200:         DOS     _PRINT                    *
201: *       addq.l  #4,sp                     *
202:
203:         clr.w   -(sp)                     *常駐終了
204:         move.l  #KEEPSIZ,-(sp)            *
205:         DOS     _KEEPPR                   *
206:
207: *
208: *       エラー終了
209: *
210: kperr:  lea.l   errms1(pc),a0             *二重に常駐しようとした
211:         moveq.l #thCONFLICT,d1            *
212:         cmp.l   d1,d0                     *
213:         beq     error
214:         *
215:         lea.l   errms2(pc),a0             *スレッド数の限界
216:         moveq.l #thNOMORE,d1              *
217:         cmp.l   d1,d0                     *
218:         beq     error                     *
219:         *
220:         lea.l   errms0(pc),a0             *processが設定されていない
221:         bra     error                     *
222: *
223: rmerr1: lea.l   errms3(pc),a0             *常駐していないのに
224:         bra     error                     *  常駐解除しようとした
```

るような記述があるが，change_prを呼び出してからつぎに実行権が返ってくるまでのあいだにすべてのスレッドに1回ずつ実行権が渡るという保証はないので，リスト4のようにchange_prとget_prをループでくくるのが正しいと思う。この場合，もし破棄要求が無視されたら無限ループに陥るが，自分の常駐部が相手なのだからリスト4ではその心配はない。自分の常駐部がときに破棄要求を無視するようになっているのであれば，繰り返し回数を設定するなり，事前に通信を交わして破棄できる体勢かどうかを確認するなりすることになるだろう。

非常駐部ではエラー処理ルーチン(210～240行)，とくに常駐時にopen_prがエラーを返したときの処理にも目を向けてほしい。open_prが失敗する可能性としては，

1)　CONFIG.SYSのPROCESSを設定していない
2)　スレッド数がすでに限界に達している
3)　同名のスレッドが存在する

が考えられるので，エラーコードに応じたエラーメッセージを出すようにしてある。手を抜いて，

> CONFIG.SYSのPROCESSが設定されていないか，スレッド数が限界に達しているか，同名のスレッドが存在します

などというエラーメッセージを出したのではCOMMAND.Xと同レベル扱いされてしまう。

20行からの常駐部をみてもらおう。21～30行が初期化部分。まず，21行ではワークエリアをアドレスレジスタ間接でアクセスできるようにするために，ワークの先頭アドレスをa6に入れている。このプログラムではワークをほとんど使わないので素直に絶対アドレスで参照するようにしてもよかったのだが，レジスタが余っているのをいいことに，こういう形にしてみた。ワークの構造は8～13行で定義してあり，その実体は82～86行にある。

1行飛ばして25～30行ではCTRL＋Cやハードウェアエラーによる中断時の処理アドレスを設定している。この設定をしないと中断時の処理アドレスは常駐処理時のままとなり，フォアグラウンドタスクを止めようとしてCTRL＋Cを押したときにたまたま制御がバックグラウンドタスク側にあったりすると，どこか変なところに飛んでいってしまう危険がある。それを避けるために，バックグラウンドタスクはかならず中断時の処理アドレスを設定しなければならない[15]。で，飛ばした22行では，中断時処理からメイン処理へ回復するときに備えてスタックポインタとワークアクセス用のレジスタを待避している。中断時処理ルーチンに制御が移ったときにはレジスタ内容が保証されないので，再設定する(76行)必要があるのだ。

32～69行がメインループだ。スレッド間通信を処理し(32～53行)，1文字表示して(55～64行)から100msほどスリープする(66～68行)という処理の繰り返しになっている。通信があったかどうかの判定には，sleep_prの戻り値は使わずにスレッド間通信バッファを直に調べる方法を採用している。

スレッド間通信の処理部は，スレッドの破棄要求とスリープ要求をサポートした標準的な作りだ。32～33行で通信の有無を確認し，通信が入っていたらコマンドを取り出し(36行)，すかさず，バッファをリセットして通信を許可する(38行)。ここで，本来は通信の処理が完了してからバッファをリセットするものだが，このプログラムではスレッド間通信バッファのデータ領域のサイズを最初から0にしてあるのでコマンド以外の付随データが送られることはなく，コマンドさえ取り出してしまえば，通信の処理中にべつの通信が入っても情報が失われる危険はない。あとは，破棄要求が送られてきたらkill_prで自殺し(44行)，スリープ要求が送られてきたらsleep_prで永久スリープに入る(50～52行)。戻ってくるはずのないkill_prの直後のbraは縁起ものだ。

1文字表示部はみてのとおりで，それが済んだら0.1秒ほどスリープして，ループする。このスリープする部分は，スリープする時間を変えたり，スリープする代わりにchange_prで実行権を放棄したり，あるいは，sleep_prもchange_prも呼ばずにループ先頭に飛んだり，といろいろ変えてみると，バックグラウンド処理に対する理解が深まるかもしれない。

さまざまなDOSコール

最後にバックグラウンド処理機能関連のそのほかのDOSコールを紹介して終わろう。

time_prはタイマ割り込みの回数をカウントしている32ビットの値を返す。間隔を空けて呼び出せば，その戻り値の差から，1ms単位での経過時間がわかる。複数のプログラムが並行動作していると，命令の実行時間に依存した方法[16]で時間を扱うことができなくなるので，べつの基準が設けられた，ということだ。

s_malloc，s_free，s_processは，これまでひと続きのものとしてのみ扱われていたメインメモリを，スレッドごとに分割して割り当てられるようにするものだ。s_processでスレッドに対してこの割り当てを行うと，以後，そのスレッドからのメモリ確保要求は割り当てられたメモリの範囲に制限される[17]。この機能は，mallocで必要に応じてメモリを確保するバックグラウンドタスクに対して，メモリを予約しておく目的で使うことになるだろう。フォアグ

```
225: *
226: rmerr2: lea.l    errms4(pc),a0            *常駐解除に失敗した
227:         bra      error                    *
228: *
229: vererr: lea.l    verrms(pc),a0            *Human68kの
230:         bra      error                    *  バージョンが違う
231: *
232: usage:  lea.l    usgmes(pc),a0            *使用法の表示
233:         *
234: error:  move.w   #STDERR,-(sp)            *メッセージを
235:         pea.l    (a0)                     *  標準エラー出力へ出力
236:         DOS      _FPUTS                   *
237: *       addq.l   #6,sp                    *
238:
239:         move.w   #1,-(sp)                 *エラー終了
240:         DOS      _EXIT2                   *
241: *
242: myname: .dc.b    'BGTEST',0               *自分のスレッド名
243: *
244: keepms: .dc.b    'BGTESTが常駐しました',CR,LF,0
245: remmes: .dc.b    'BGTESTを切り離しました',CR,LF,0
246: *
247: errms0: .dc.b    'CONFIG.SYSのPROCESSが'
248:         .dc.b    '設定されていないようです',CR,LF,0
249: errms1: .dc.b    'BGTESTはすでに常駐しています',CR,LF,0
250: errms2: .dc.b    'これ以上バックグラウンドプロセスを'
251:         .dc.b    '起動できません',CR,LF,0
252: errms3: .dc.b    'BGTESTはまだ組み込まれていません',CR,LF,0
253: errms4: .dc.b    'BGTESTが常駐解除できません',CR,LF,0
254: verrms: .dc.b    'Human68kのバージョンが違います',CR,LF,0
255: *
256: usgmes: .dc.b    '機　能: バックグラウンドプロセスのテスト',CR,LF
257:         .dc.b    '使用法: BGTEST [/R]',CR,LF
258:         .dc.b    TAB,'/R',TAB,'常駐解除する',CR,LF,0
259: *
260:         .bss
261:         .even
262: *
263: thinfo: .ds.b    SIZEofTHREADINFO         *スレッド情報格納領域
264: *
265:         .stack
266:         .even
267: *
268:         .ds.l    2048
269: inisp:
270:
271:         .end     ent
```

ラウンドで走るプログラムの起動時には最大のメモリブロックが割り当てられるので，いざバックグラウンドタスクがメモリを確保しようとしたときに使えるメモリがないことも考えられる。そこで，あらかじめs_processでメインメモリを切り出して，そのスレッド専用に割りつけておくというわけだ。で，s_malloc，s_freeは無条件にメインスレッドにわりあてられたメモリに対してメモリブロックの確保/解放を行う。

malloc2はDOSコールmallocをメモリ確保の方法が指定できるよう改良した上位コールだ。mallocはメモリの確保を無条件に若いアドレス側から行ったが，malloc2ではモードを指定することで，逆にメモリの高位側から確保したり(モード2)，あるいは，フリーメモリがいくつかのブロックに分割されているときに要求サイズを満たす最小のブロックからメモリを割り当てたり(モード1)といったことができる。バックグラウンド処理と直接の関連はないが，バッチプログラム中でプログラムを常駐させる際にメモリの分断を防ぐ目的で自身をメモリの最高位に移動したりする場合には有効だろう。

あとは，execがサポートするようになったオーバーレイXファイルだ。これもバックグラウンド処理専用というわけではないが，常駐部と非常駐部をべつのXファイルにしてある場合などにはそれなり

15) 複数タスクからなるバックグラウンドプロセスの場合，中断時の飛び先は共用される。

16) たとえば，dbraで何回空ループを実行すると何msである，とか。

17) ちなみに，Human68k内部でのスレッド管理情報は，get_prが返すものよりも8バイト大きく，この部分にs_processによって割り当てられたメモリについての情報が格納されている。

に役に立つ。ここでいうオーバーレイXファイルは複数のXファイルをBIND.Xでひとまとめにしたものを指す。オーバーレイXファイルは，単純にXファイルを連結し，末尾にモジュールリストを付け加えた格好をしている。モジュール一覧の位置は，先頭モジュールのXファイルヘッダに格納される。参考までにXファイルのヘッダ構造を表2に示しておく。興味のある人は適当にXファイルをダンプして，見比べてみるといいだろう。ここでは，03H バイト目のロードモードについてだけ触れておく。

比較的最近まで気づかなかったのだが，このバイトの第1ビットを立てておくと，テキストセクションからスタックセクションまでがちょうど収まるだけのメモリブロックがメモリの高位アドレス側から確保され，プログラムはそこにロードされるようになる。Human68k ver 2.0でのマイナーな拡張機能だ。メモリの高位に常駐部を置くプログラムでは便利な機能だろう。なお，どうせ隠れオプションがあるだろうと，いまLK.Xで試してみたら/Aオプションでロードモードの設定ができるようだ。/Aの直後には立てたいビット番号を指定する。0から7までが有効だが，現在意味があるのは1だけだ。

ところで，第1ビットというのがどうも中途半端だが，どうも第0ビットにも意味を持たせ，このビットが1だったらmalloc2のモード1相当の方法でメモリを割り当てるつもりだったらしい。ところが，Human68kのこの判定部分がバグって意味がなくなってしまったようだ。

＊　＊　＊

バックグラウンド処理は使いようによっては便利だし，遊びがいもある(こっちのほうが肝心だな)。複数プログラムからのコンピュータ資源の共有については少々難があり，バックグラウンドタスクが利用できる(してもよい)資源に制約がある[18]が，その点に目をつぶればマルチタスクっぽさも味わえる。

それにしても，前回，今回と，手元の解析メモから未公開情報を引っ張り出してみたわけだが，我ながら，ずいぶん無駄なことに時間を使っているものだと途中で結構情けなくなった。来年はもっと有意義に時間を使いたいものだ，と，2月号であることをすっかり忘れて実時間に引き戻されそうになる。

次回からは文字列の探索をとり上げる。たぶん，来月はふつうの文字列探索で，再来月はその一歩先，へたするともう1回費やしてそのまた一歩先までいくかもしれない。来月には間に合わないだろうが，クレームをつけるならいまのうちだ。

表2　Xファイルヘッダの構造

00H	2b	識別ID('HU'=48h 55h)
02H	1B	(予約?)
03H	1B	ロードモード(bit 1 = 1のとき高位アドレスにロード)
04H	1L	ベースアドレス
08H	1L	実行開始アドレス
0CH	1L	テキストセクションサイズ
10H	1L	データセクションサイズ
14H	1L	ブロックストレージセクションサイズ(.comm, .stackを含む)
18H	1L	再配置情報サイズ
1CH	1L	シンボルテーブルサイズ
20H	1L	行情報サイズ(SCD.X用)
24H	1L	拡張シンボルテーブルサイズ(SCD.X用)
28H	1L	拡張シンボルテーブル2サイズ(SCD.X用)
2CH	4L	(予約)
3CH	1L	バインドされたモジュールリストのファイル先頭からの位置

18) キーを押したときにどのスレッドが実行されているか予測できないので，キーボードはほとんど使えないし，当然，画面表示はフォアグラウンドタスクの邪魔をしないように注意しなければならない。マウスやジョイスティックもつねに使えるという保証はないし，バックグラウンドタスク側から使ってもよいかどうかお伺いを立てる方法もない。

バックグラウンド機能のパラメータ設定

バックグラウンド機能を利用するためには，CONFIG.SYSのPROCESS行を設定しておく必要がある。ここで，設定時の注意点を簡単にまとめておきたい。

PROCESSには，最大スレッド数，メインスレッドの実行優先レベル，タイムスライス値，の3引数を与える。最大スレッド数は言葉どおりで，同時に走らせるスレッド数の最大値を2～32の範囲で指定する。起動時にはこの数の分だけスレッド管理情報格納用のメモリ(124バイト/スレッド)が確保される。メインスレッドが無条件に用意されるため，実際に走らせることのできるバックグラウンドタスクの数は設定値よりも1小さい。

第2引数のメインスレッドの実行優先レベルもそのまんま。本文で触れたように，有効な範囲は2～255で，値大きいほどレベルが高い。通常，何も考えずに最高レベルの2にしておけばよいだろう。

第3引数のタイムスライス値はスレッドを切り替える時間間隔(本文には出てこなかったが，これをタイムスライスという)を表す。有効な設定値は1ms単位で1～100だ。タイムスライス値はあまり極端に小さな値にはしないほうがよいだろう。スレッド切り替えにだって時間はかかるから，頻繁にスレッドを切り替えればそれだけ無駄になる時間も増える。また，プログラムの作り方に問題があるといってしまえばそれまでなのだが，一連の処理を行ってはしばらくスリープして待機するタイプのバックグラウンドタスクは，起きてからスリープするまでのあいだにスレッドが切り替わらない程度のまとまった時間を与えないと，ときに誤動作する場合がある。といって，タイムスライス値を大きくしすぎると，今度はスリープせずに与えられた時間を目一杯使うバックグラウンドタスクを走らせたときに，スレッド切り替えの粗さが目立つようになり，精神衛生上よくない。マニュアルに設定例として示されている20msぐらいがほぼ適正値だろう。もっとも，使用するプログラムによっても状況は変わってくるから，あれこれ試してみて最適な値を探すに越したことはない。

ちなみに，タイムスライス値を微調整する際にいちいちCONFIG.SYSを書き換えて再起動するのが面倒だという悪い子は，デバッガで以下のアドレスをいじるのもいいかもしれない。

Ver.2.01　E8A1H
Ver.2.02　E923H
Ver.2.03　E6AFH

バックナンバー案内

BACK ISSUES

ここには1992年2月号から1993年1月号までをご紹介しました。現在1991年1,5,8,9,11,12,1992年1,6～12,1993年1月号の在庫がございます。バックナンバーおよび定期購読の申し込み方法については162ページを参照してください。

1992

2月号(品切れ)
特集　2Dグラフィックの拡張

連載　響子 in CGわ～るど/大人のためのX68000/マシン語プログラミング
ハードエ作/ショートプロ/ANOTHER CG WORLD/Z80's Bar
吾輩はX68000である/Computer Music入門/カードゲーム

●TREND ANALYSIS
●MIRAGE Model Stuff/Press Conductor PRO-68K
LIVE in '92　ストリートファイターII/Tide Over
THE SOFTOUCH　ジェノサイド2/アルシャーク/コード・ゼロ 他
全機種共通システム　シミュレーションゲームPOLANYI

3月号(品切れ)
特集　SCSIの活用

連載　響子 in CGわ～るど/DōGA CGA/大人のためのX68000/Z80's Bar
ショートプロ/吾輩はX68000である/マシン語プログラミング
ハードエ作/ANOTHER CG WORLD/Computer Music入門/カードゲーム

●Z-MUSIC支援ツール　ZPDCON.X
●Z's-EX用拡張コマンド　MASK_reverse.X
LIVE in '92　ギャラクシーフォース/君が代
THE SOFTOUCH　グラディウスII/レミングス/大戦略III'90/伊忍道
全機種共通システム　カードゲームKLONDIKE

4月号(品切れ)
特集　成熟するゲームと日本の文化

連載　よい子のSX-WINDOW/Z80's Bar
響子 in CGわ～るど/ショートプロ/吾輩はX68000である
ハードエ作/ANOTHER CG WORLD/Computer Music入門

●発表　1991年度GAME OF THE YEAR
●バーコードバトラー
LIVE in '92　あじさいのうた/ショパン練習曲作品25-2へ短調/IT'S MAGIC
THE SOFTOUCH　ファーストクィーンII/マスターオブモンスターズII 他
全機種共通システム　実践Small-C(1)オプティマイザO80

5月号(品切れ)
特集　明日のための環境づくり
第7回　言わせてくれなくちゃだワ

連載　響子 in CGわ～るど/大人のためのX68000/Z80's Bar
ハードエ作/ショートプロ/マシン語プログラミング
Computer Music入門/吾輩はX68000である

●製品紹介　MIDI音源 03R/W/MIC68K
LIVE in '92　フレンズ/Danger Line
THE SOFTOUCH　エイリアンシンドローム/苦胃頭捕物帳 他
全機種共通システム　実践Small-C(2)COMMAND.OBJ

6月号
特別企画　Oh!MZ,Oh!X10年間の歩み
特別付録　創刊10周年記念PRO-68K(5"2HD)

連載　響子 in CGわ～るど/大人のためのX68000/マシン語プログラミング
ハードエ作/ショートプロ/ANOTHER CG WORLD/Z80's Bar
吾輩はX68000である/Computer Music入門

●新製品紹介　Z'sSTAFF PRO-68K ver.3.0
LIVE in '92　Shake the Street/Ancient relics
THE SOFTOUCH　スピンディジーII/ロイヤルブラッド/ライフ&デス 他
全機種共通システム　実践Small-C講座(3)COMMAND.OBJ2

7月号
特集　超空間美術論
特別付録　DōGA CGAシステム&お試しディスク(5"2HD)

連載　よいこのSX-WINDOW/響子 in CGわ～るど/Z80's Bar
ANOTHER CG WORLD/大人のためのX68000
Computer Music入門/ハードエ作/ショートプロ

●試用レポート　V70アクセラレータボード
LIVE in '92　Bye Bye My Love/MATERIAL GIRL/ヴェクザシオン
THE SOFTOUCH　将棋聖天&棋太平68K/シムアース/太閤立志伝
全機種共通システム　実践Small-C講座(4)関数リファレンス

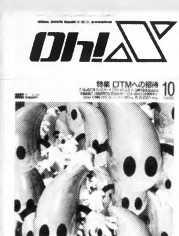
8月号
特集　プログラミング再入門

連載　響子 in CGわ～るど/吾輩はX68000である/よいこのSX-WINDOW
マシン語プログラミング/ハードエ作/ANOTHER CG WORLD
大人のためのX68000/Computer Music入門/ショートプロ

●新製品紹介　MATIER/TG100/SOUND SX-68K
LIVE in '92　氷穴/ガラガラヘビがやってくる/風の贈り物
THE SOFTOUCH　三國志III/シムアース/ウルティマVI/バトルテック
全機種共通システム　実践Small-C講座(5)ワイルドカード
グラフィックライブラリGRAPH.LIB

9月号
特集　数値演算の熱い逆襲

連載　DōGA CGアニメーション講座/大人のためのX68000
響子 in CGわ～るど/吾輩はX68000である/ショートプロ
マシン語プログラミング/ハードエ作/ANOTHER CG WORLD

●新製品紹介　MATIER/MIREGE Model Stuff
LIVE in '92　恋をしようよ Yeah! Yeah!/ゆめいっぱい
THE SOFTOUCH　ファイナルファイト/ライジングサン/
ヨーロッパ戦線/シューティング68K GAMES
全機種共通システム　O-EDIT & MODCNV

10月号
特集　DTMへの招待

連載　DōGA CGアニメーション講座/大人のためのX68000
響子 in CGわ～るど/吾輩はX68000である/ショートプロ
マシン語プログラミング/ハードエ作/ANOTHER CG WORLD

●試用レポート　X68000用CD-ROMドライブ
LIVE in '92　美少女戦士セーラームーン/笑顔を探して 他
THE SOFTOUCH　ポピュラスII/リーディングカンパニー/
ネクタリス/サークII
全機種共通システム　実践Small-C講座(6)SLENDER HUL

11月号
特集　ゲームマネージメント

連載　DōGA CGアニメーション講座/大人のためのX68000
響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW
ハードエ作/ANOTHER CG WORLD/Computer Music入門

●新製品紹介　CHART PRO-68K
LIVE in '92　ストリートファイターII/スーパーマリオ 他
THE SOFTOUCH　キャッスルズ/シュートレンジ/
ポピュラスII/サンダーレスキュー
全機種共通システム　実践Small-C講座(7)EDIT

12月号
Oh!X5周年特別企画　ショートプロ大集合

連載　DōGA CGアニメーション講座/マシン語プログラミング/
響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW
大人のためのX68000/ハードエ作/Computer Music入門

●エレクトロニクスショウ'92
LIVE in '92　LAST CHRISTMAS/闇の血族/ユーフォリー
THE SOFTOUCH　デスブレイド/ムーンクレスタ&テラクレスタ/
ふしぎの海のナディア/ロードス島戦記II 他
全機種共通システム　実践Small-C講座(8)MAKE

1993

1月号
特集　D.I.Y.ハードウェア

連載　DōGA CGアニメーション講座/マシン語プログラミング/
響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW
大人のためのX68000/ハードエ作/Computer Music入門

●新製品紹介　サンダーワード/SX広辞苑
LIVE in '92　ムーンライト伝説/チャコの海岸物語
THE SOFTOUCH　オーバーテイク/ストライダー飛竜/
エアーマネジメント/パイプドリーム 他
全機種共通システム　実践Small-C講座(9)EDC-Tの拡張

アクセラレータを作る（その2）

GALの概要とソフトウェア互換性

Ishigami Tatsuya 石上 達也

先月発表した基本回路図の変更と回路の構成に使用するGAL素子について解説します。また，68020への変更の際障害となるソフトウェアの問題とその回避方法についても解説します。

先月号で，これからの予定というものを書いたような気がするのですが，さっそく今月は予定どおりではありません。2，3月号に予定していた分をいっぺんにやってしまいます。

前回，68000と68020との違いを説明しましたが，そのほとんどが68000は6800系のLSIをつなげるように工夫がなされていたのに対して，68020にはそのような機能はない，というものでした。そして，68000の代わりに68020を差してみようというのがアクセラレータでしたから，製作のメインは6800系LSI用の信号をどうやって作るかということになります。

さて，編集部からいろいろ遊んでいいよ，とX68000EXPERTをいただいて，いろいろ遊んでいると，いろいろなことがわかりました。

X68000のケースを開ければ68XXというLSIがないことくらいはわかるのですが，68000に80系あるいは86系のLSIをつなげるときにこれらの信号を使う人がいたりするので一応チェックしてみました。

●実験報告（その1）

拡張スロットのA34（$\overline{\mathrm{VMA}}$）に図1のような回路をつないでみました。これは一瞬でも$\overline{\mathrm{VMA}}$がアサートされるとLEDが点灯するというものです。しかし，OPMやPCMを鳴らしたり，ディスクを読み書きしたり，画面表示をさせてみたりしてもLED は点灯しませんでした。つまり，この信号はアサートされないということです。

図1　$\overline{\mathrm{VMA}}$がアサートされるかどうかの判定回路

●実験報告（その2）

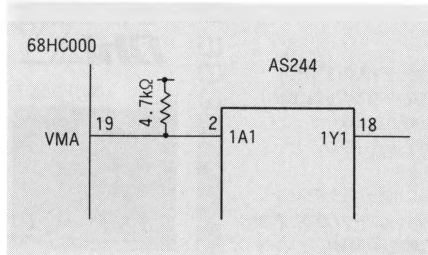
拡張スロットに出ている信号がアサートされないからといって油断はできません。次に，68000の19番ピン（$\overline{\mathrm{VMA}}$）を引き抜いてみました。$\overline{\mathrm{VMA}}$周りの配線をざっと調べたところ図2のような感じでしたから，このピンが接続されていないと信号はずっとハイレベルになったままになるわけで，X68000内部の回路にも$\overline{\mathrm{VMA}}$がアサートされたという情報は伝わりません。それでもX68000はきちんと動作しました。

●実験報告（その3）

調子にのって，その隣のEクロック出力を抜いてみました。抜いてもX68000はきちんと動きました。もしかしたら$\overline{\mathrm{BERR}}$を発生するタイミングの作成（$\overline{\mathrm{DTACK}}$がタイムアウトした場合の処理）にこのEクロックを使っているのではないかと思いましたが，大丈夫なようです。DB.Xで実装されていないメモリエリアをアクセスしようとすると，ちゃんと（といっていいものかどうか）バスエラーが発生しました。

●結論

$\overline{\mathrm{VMA}}$がアサートされないで，しかも接続をはずしてしまってもまったく問題が起こらなかったということは，この信号がまったく参照されていないということで，

図2　68HC000の$\overline{\mathrm{VMA}}$周辺

$\overline{\mathrm{VPA}}$が6800系LSIのアクセスを始めるために使用されていないということです。つまり，$\overline{\mathrm{VPA}}$はすべてオートベクタ割り込みの要求に使用されているということにほかなりません。

Eクロックをはずしてしまってもまったく問題が起こらなかったということも考えあわせると，X68000は6800系LSIをアクセスするサイクルにはならないということがいえます。

というわけで，Eクロックはいらないし，$\overline{\mathrm{VPA}}$は$\overline{\mathrm{AVEC}}$に直結すればいいし，となってアクセラレータの回路は随分と簡単になりました。図3に新しい回路図を示します。

GALについて

以上のように，この連載ではあっちを変更したり，こっちを変更したりというようなことが毎月のように起こります。先月号の写真からもわかるように，アクセラレータのボードはとても高密度な配線を行います。変更のたびにハンダゴテを取り出してきて配線を変えていては，汚くなってしまいますし，変更のたびにどこかの配線を余計に切ってしまったりというようなことが起こらないとも限りません。奥まったところの配線を変更するのに，手前の配線をいったん取り外さなければならないというようなこともあるでしょう。

そんなわけで，この連載ではGALを使用します。GALの大まかな内容については先月号の囲みを参照してください。

最終回が待てずに，先月号の回路をいきなり組みあげてしまった人もいるかもしれませんが，大丈夫です。20V8に接続されているFC0～1やA16～19を無視するだけですから。GALにしておいてよかったでしょう。

さて，そのGALの働きを記述するのにGALコンパイラというものが必要になっ

てきます。コンパイラというのはCコンパイラとかPASCALコンパイラというときと同じで，翻訳機という意味です。今回はそのコンパイラにK.E.M Electronics Ltdという会社のmini-CUPLというものを使用しました。これは，LOGICAL DEVICES,INCのCUPLのサブセットで，対象となる半導体を16V8と20V8に制限したものです。サブセットといっても，この2種類のGALを使っている分にはなんら問題はありません。

たいていのGALコンパイラがそうであるように，このmini-CUPLもIBM PC上で動作します。PC-9801上でも一応は動作するのですが，回路シミュレータなどのグラフィック描画関係がうまくいかないようです。

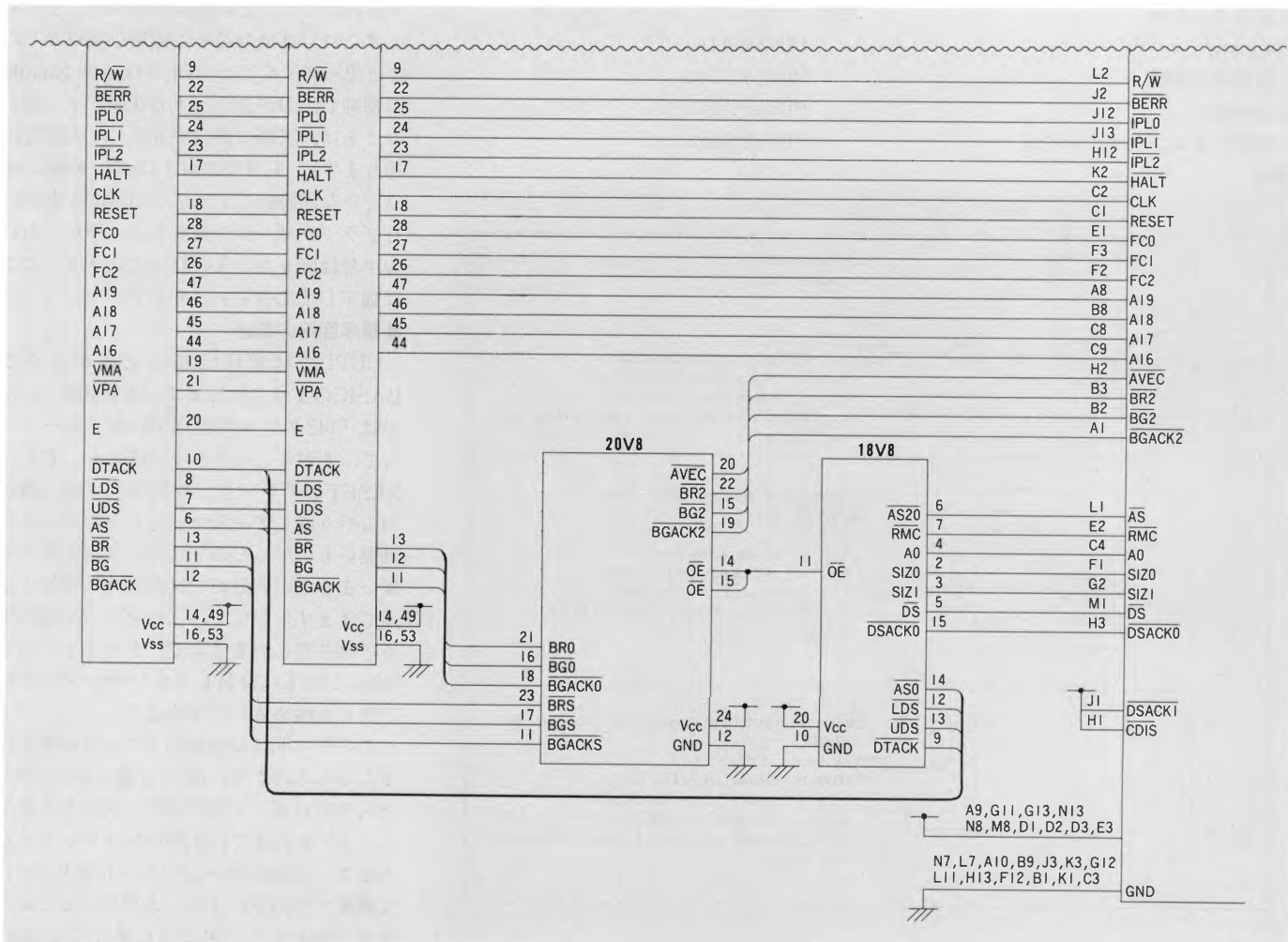
X68000用のアクセラレータを作ろうとしていて，IBM PCを持っていて，しかもmini-CUPLを持っている人となると，かなり限られてくるでしょう。そこで，この連載ではGALコンパイラのソースファイル（以下では，PLDファイルという）を掲載するときには，そのコンパイル結果のヒューズパターン（以下，JEDECファイル。Joint Electron Devices Engineering Council:「半導体製造業者の公的規約会議」の略らしい。Cコンパイラなどでいうところの，オブジェクトファイル）を掲載します。このヒューズパターンというのは，どのコンパイラを用いても，GALに焼き込む内容が同一になるなら同一になります。つまり，ヒューズパターンを入力すれば，PC-9801とそれにつながるGALライタを持っている人ならば，これを利用できるわけです。

GALライタ単体であればPC-9801用のものが，いろいろと雑誌に発表されているので安価に製作できそうです。私個人としては，参考文献1がおすすめです。これはプリンタに接続するタイプのもので，製作も簡単で，だいたい5000円くらいで製作できそうです。なお，参考文献2にこれをAバージョン（とりあえず，ノーマルバージョンの高速版だと思ってください。アクセラレータの製作ではこちらを使うことを奨励します）に対応させたものが出ているので，これもあわせて調べてみるとよいでしょう。

mini-CUPLについて

Oh!Xに掲載されたオブジェクトプログラムのダンプリストを眺めても，さっぱりわけがわからないように，GALのJEDECファイルを眺めても内容がわかる人はいません（と，いい切ってよいような，悪いような……よくわからない雑誌なんだな，Oh!Xって）。いちいち等価回路を示していくのも面倒ですし，GALを使うメリットが消えてしまいます。

そんなわけで，PLDファイル（GALのソースファイルのこと）の読み方を説明します。この説明は掲載するリストを読めるようになるためのものです。これさえマスターすればGALを自由に焼き込めるというものではありません。しかし，「読める」ようになれば，ほかのコンパイラ（たとえば，PALASMの新しいバージョンだとか，PAGASMとか，OrCADだとか）へ移植し

図3　新しい回路図（上部は先月と同じ）

たりできるようになりますから，社会人の方や電気系の大学生の方は，周囲をあさってみて利用できる環境を最大限に活用するのもひとつの手かもしれません。

PLDファイル自体はただのテキストファイルです。ここらへんはCコンパイラと同じです。好きなエディタを用いて好きなように書きます。あまり勝手な書き方をするとコンパイラが文句をいいます。

●ヘッダ

PLDファイルの先頭には，以下のような項目があります。これらを省略するとコンパイラが受け付けてくれません。項目の終わりには“；”を入れ，セパレータとします。

Name:
　GALに名前をつけ，ここに書きます

PartNo:
　パートナンバーを記述します

Date:
　日付を書きます

Revision:
　リビジョン（何回手直ししたかとか）

Designer:
　設計者の名前

Company:
　設計者の所属

Assembly:
　使用するコンパイラの名前

Location:
　基板のどこに配置するGALなのか（U1とかU2とか基板にシルク印刷する場合はそれにあわせる）

これ以外の項目を記述したいときには，「/＊～～＊/」で囲まれた範囲が注釈とみなされるので，ここに記述を行います。

●ピン配置

ヘッダの次には必ず，GALのピン配置を記述します。ピンの名前はNC(Non Conected：接続しない）以外に同じものを使用してはいけません。

例）　PIN 1 = CLOCK;

1番ピンをCLOCKと名づける（名づけただけで，ここではこの信号がはたしてどのように入力端子，あるいは出力端子として機能するのかは問われない）。

信号が負理論（0Vのときに有効で，5Vのときに無効なもの）の場合には，名前の前に「！」をつけます。

例）　PIN 2 = !RESET;

2番ピンを/RESETと名づける。

また，省略形として以下のようなものがあります。

例）　PIN {2..5} = {A1..4}；
　　PIN2をA1に，
　　PIN3をA2に，
　　PIN4をA3に，
　　PIN5をA4に，

のように対応づける。

図4

●等価式の記述（組み合わせ回路）

ピンの配置および命名が済んだら，いよいよその機能を定義していきます。といってもそんなに難しい決まりごとがあるわけではなくC言語の方言みたいなものです。ここで使える演算子には以下の5種類があります。

=	代入
!	反転
#	論理和（OR）
&	論理積（AND）
$	排他的論理和（EX-OR）

このとき，代入の左に現れた名前で示されるピンが出力ピンとなります。基本的には右が入力ピンですが，出力ピンの内容をフィードバックさせることもできます。さらに，ピンに出さない一時的な変数を使うこともできます。

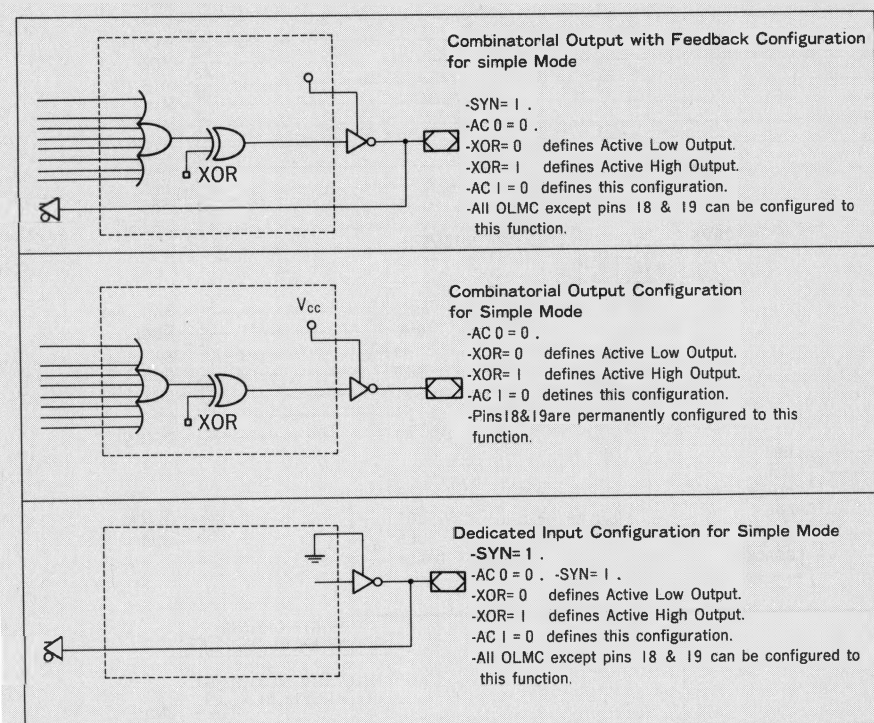
例）　TEMP = AND1 & AND2；
　　ANDOUT = TEMP & AND3；

AND1とAND2とAND3の論理積をANDOUTに出力する。

あと，左辺に「.OE」とか「.D」とか余計なものがつくことがあるのですが，とりあえずこれらは出力端子の種類を決めるものだと思ってください（.OEがOutput Enableの制御で，.DがDラッチの制御です。細かいことは，実際に使用する際にその都度説明します）。先月号の図5（あの，蜘蛛の巣みたいな回路図です）で，OLMCと書かれたブラックボックスがありましたが，あれの中身は図4のようになっています。ここに細工して出力をいじります。

●順序回路の記述

CUPLでは順序回路の記述がまるでBASICのように行えます。順序回路というのは「142クロック後にPIN3をアサートにして，それから11クロック後に～。でも，RESETがアサートされた場合には，最初からやり直して～云々」というたいへん面倒臭いものだったのですが，CUPLを使えばいままでの苦労がウソのように簡単に記述できます。ただし，これをすべて説明するのは非常に大変なので，リスト1に示すmini-CUPLに付属していたサンプルプログラムを眺めるにとどめます。

このリストは10進数のカウンタ回路を作るためのものです。dir（3番ピン）に0Vを入力すれば，アップカウンタになりますし，5Vを入力すればダウンカウンタになります。結果はQ0～Q3（17～14番ピン）に2進数で出力されます。カウントしている最中で桁あふれが起こった場合にはcarry

（18番ピン）をアサートします。clr（2番ピン）に5Vを入力すれば，どんな状態であっても，カウントを再び0から始めます。クロック入力はclk（1番ピン）に加えます。ここには書いてありませんが，GALへ入力するクロックはすべて立ち上がり時にトリガーします。

以上を踏まえて，リスト1を「眺めて」いきます。ヘッダとピンの定義はいままで述べてきたとおりです。34行目の，

field count = {Q3..0}；

というので，Q0～Q3を“count”という名前のカタマリとして扱うと宣言しています。次の35行からは，プリプロセッサ命令です。C言語の#define命令とほぼ同じです。

45行目ではclrとdirを” mode”という名前のカタマリとして扱うと宣言しています。この組み合わせが，

clr = 0，dir = 0

のときにはupが有効，

clr = 0，dir = 1

のときにはdownが有効，

clr = 1，dir = 0

のとき，または，

clr = 1，dir = 1

のときにはclearが有効となるように，次の46～48行目で定義されています。

以上のようにやってきて，やっと順序回路の記述に入れます。50行目の，

sequence count {

というので，「ここでは，countについて話を進めるよー」と宣言しています。前述のように，countというのは，Q0～Q3のことでしたから，これらについて話を始めるわけです。次の行から，

present S0

のような記述がありますが，これが「ラベル」に相当します。S0というのは38行目で’b’0000と同じだよ，と宣言されているのでした。

いままでの話をまとめると，出力Q0～Q3がすべて0のときには，このpresent S0のところに記述されているものをクロック入力時に実行するのです。Q0が1で，Q1～Q3が0の場合にはpresents 1のところを実行します。

実行する中身のif ～というのは，C言語などと同じで，「もし～なら」という構文です。present S0では，

・upが有効なときには，次にpresent S1へ行け（アップカウンタとして作動せよ）
・downが有効なときには，次にpresent S9へ行け（ダウンカウンタとして作動せよ）
・clearが有効なときには，次にpresent S0へ行け（初期化して最初の状態に戻れ）

というように命令されています。ここらへんの「ノリ」は非常に重要です。ほかの「ラベル」のところにも，同じような記述がなされています。ただ例外的に，最後の行では（つまりpresent S9）では，

out carry

という記述が見受けられます。状態S9になったということは，桁あふれが生じたということなので（アップカウンタの場合には繰り上がって10になるし，ダウンカウンタの場合には繰り下がりが起こったことになる），carryを出力するようにしています。

細かいことをいえば「電源投入時には，出力が0～9のあいだにいるかわからないから，S10～S15も安全性のために必要なんだよ」とか，いろいろいいたいことがあるのですが，それはまたの機会にしましょう。

リスト1

```
 1: Name      Count10;
 2: Partno    CA0018;
 3: Date      12/19/89;
 4: Revision  02;
 5: Designer  Kahl;
 6: Company   Logical Devices, Inc.;
 7: Assembly  None;
 8: Location  None;
 9: Device    g16v8a;
10:
11: /******************************************************************/
12: /*                                                                */
13: /* Decade Counter                                                 */
14: /*                                                                */
15: /* This is a 4-bit up/down decade counter with synchronous        */
16: /* clear capability.  An asynchronous ripple carry output is      */
17: /* provided for cascading multiple devices.  CUPL state machine   */
18: /* syntax is used.                                                */
19: /******************************************************************/
20: /* Allowable Target Device Types :  PAL16RP4, GAL16V8, EP300      */
21: /******************************************************************/
22:
23: /**  Inputs  **/
24:
25: pin 1      = clk;            /* Counter clock               */
26: pin 2      = clr;            /* Counter clear input         */
27: pin 3      = dir;            /* Counter direction input     */
28: pin 11     = !oe;            /* Register output enable      */
29:
30: /**  Outputs  **/
31:
32: pin [14..17] = [Q3..0];      /* Counter outputs             */
33: pin 18 = carry;              /* Ripple carry out            */
34:
35: /** Declarations and Intermediate Variable Definitions **/
36:
37: field count = [Q3..0];       /* declare counter bit field */
38: $define S0 'b'0000           /* define counter states */
39: $define S1 'b'0001
40: $define S2 'b'0010
41: $define S3 'b'0011
42: $define S4 'b'0100
43: $define S5 'b'0101
44: $define S6 'b'0110
45: $define S7 'b'0111
46: $define S8 'b'1000
47: $define S9 'b'1001
48:
49: field mode = [clr,dir];      /* declare mode control field */
50: up = mode:0;                 /* define count up mode */
51: down = mode:1;               /* define count down mode */
52: clear = mode:[2..3];         /* define count clear mode */
53:
54: /** Logic Equations **/
55:
56: sequence count {                       /* free running counter */
57:
58: present S0      if up        next S1;
59:                 if down      next S9;
60:                 if clear     next S0;
61:                 if down      out  carry;
62: present S1      if up        next S2;
63:                 if down      next S0;
64:                 if clear     next S0;
65: present S2      if up        next S3;
66:                 if down      next S1;
67:                 if clear     next S0;
68: present S3      if up        next S4;
69:                 if down      next S2;
70:                 if clear     next S0;
71: present S4      if up        next S5;
72:                 if down      next S3;
73:                 if clear     next S0;
74: present S5      if up        next S6;
75:                 if down      next S4;
76:                 if clear     next S0;
77: present S6      if up        next S7;
78:                 if down      next S5;
79:                 if clear     next S0;
80: present S7      if up        next S8;
81:                 if down      next S6;
82:                 if clear     next S0;
83: present S8      if up        next S9;
84:                 if down      next S7;
85:                 if clear     next S0;
86: present S9      if up        next S0;
87:                 if down      next S8;
88:                 if clear     next S0;
89:                 if up        out carry;       /* assert carry output */
90: }
```

ソフトウェア編
SOFTWARE

68000と68020の相違点

モトローラの68020のマニュアルを見ていると，オブジェクトレベルでの68000との完全互換性を謳ってあるのですが，現実はそんなに甘くはありません。どちらかというと，68000にはやや思想的な欠陥があり，これで本格的なマルチタスクOSを走らせようとすると，つらい作業となってしまいます。68010以上のMPUをホンモノにするための，やむない処置だとは思いますが，この時点で68000とのソフトウェア的な相違点が生じてしまいました。もっとも，ユーザープログラムを組んでいる場合には，それほど意識しなくても済むのですが，ことスーパーバイザモードで動作する（正確にはこのモードへ移行する）プログラムを書くときには，これらの相違点に注意をしなくてはなりません。

具体的には，68020は以下の２点で68000と異なります。

●MOVE　SR, Dnが特権命令となった
●例外処理におけるスタックの深さが違う

解決法は，まず，

MOVE　SR, Dn

が特権命令となったことに関する解決方法です。少々乱暴ですが，ここではMPUが，

MOVE　SR, Dn

を実行しようとしたら，その命令を無条件に，

MOVE CCR, Dn

に書き換えてしまうのです。具体的には，

MOVE　SR, Dn

を68020のユーザモードで実行しようとすると，例外処理の８番（特権違反）が発生します。そこで，この例外処理の先頭に，

```
        movem.l  d0/a0,-(a7)
        movem.l  10(a7),a0
        move     (a0),d0
         *実行しようとした不当命令
        andi     #$ffc0,d0
        cmpi     #$40c0,d0
        moveSR,  Dnか？
        bne      skip
        ori      #$200,(a0)
         *move CCR,Dnに書き換える
        moveq    #9,d0
        movec    d0,cacr
         *キャッシュをクリア
        move     (a7)+,d0/a0
        rte
skip:   move     (a7)+,d0/a0
          ⋮
        以下，本来の処理
          ⋮
          ⋮
```

などの，プログラムをもぐり込ませておくのです。この処理の中で，なぜキャッシュを書き換えるかというと，そりゃプログラムの自己書き換えに走ったのですから，メモリ上のデータと，プログラムキャッシュ上との同一性（コヒーレンス）を保つためにやむをえないのです。そう，高速グラフィック描画パッケージMAGICなどの，自己書き換えテクニックを使っているようなプログラムに対しても，アクセラレータは無力です。無力というよりも，キャッシュを有効にしていた場合には確実に誤動作します。ここでは，そのようなときにはキャッシュを無効にして，世の中とはそんなものと素直に割り切りましょう。

さて，より根本的な問題を含んでいるのが２番目の相違点，すなわちスタックの深さが違うことです。具体的には，図５を見てください。68000で例外処理に移行する場合，その例外処理の内容がどうであれ，図5-aのようにしかスタックは積まれませんでした。よって，バスエラーが一度発生すると，どのアドレスのメモリアクセスに対してエラーが発生したのかがわからずに，それ以前の状態を完全に復活させることは非常に困難な作業でした。このことが，68000で仮想記憶機能をサポートしようとする際，一番のネックになっていました。68020では仮想記憶と引き替えに，この点における68000との互換性を失ってしまったのです。バスエラーに限らず，例外処理ではとにかく図5-b, c，d，eです。

これがX68000において害をもたらすのがHumanのDOSコールや，SX-WINDOWのファンクションコールを利用する際です。ご存じのように，これらのファンクションコールは引数をすべてスタック上に積むことによってシステムに渡しています。つまり，この内容をa7レジスタ相対で参照している箇所はみな，その相対値が狂ってしまうのです。この相対値を再調整してやるか，それとも処理前にスタックの内容を調整してやるか。道は２つあります。

Macintoshのアクセラレータでは前者がメジャーな方法のようですが，今回は後者を採ろうと思います。相対値をすべて調節してやるというのは，HumanやFSXをすべて解析しなければいけないわけで，私ひ

図5-b　68000/68008のバスエラーとアドレスエラーに対するスタックフレーム

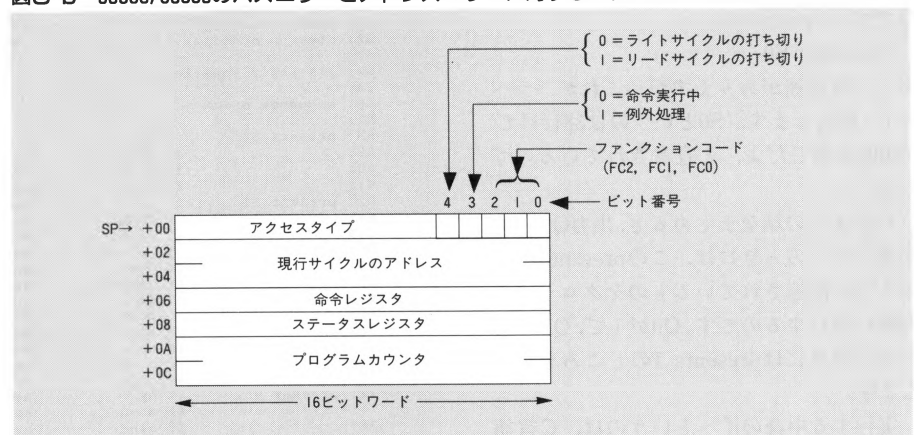



図5-a　68000/68008のショートスタックフレーム

オフセット	内容
SP→ +00	ステータスレジスタ
+02	プログラムカウンタ
+04	

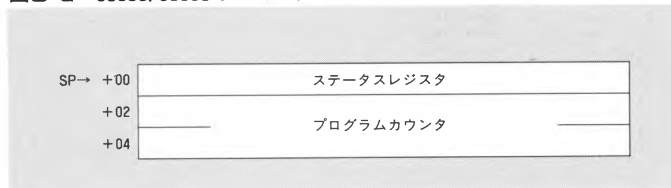
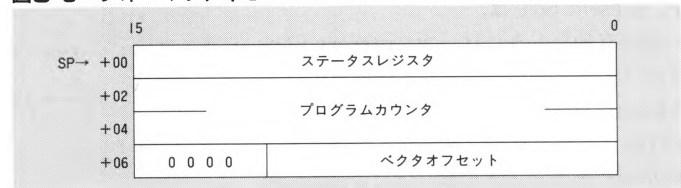



図5-c　フォーマット$0のスタックフレーム（4ワード）

オフセット	15	0
SP→ +00	ステータスレジスタ	
+02	プログラムカウンタ	
+04		
+06	0 0 0 0	ベクタオフセット

とりの力ではとても及ばないからです。ちなみにIOCSコールは引数をすべてレジスタで渡すため，変更の必要はありません(ただし，空ループでタイミングを取っているところは変更しなければいけないかもしれません。そこらへんのところは，クロックアップ組からノウハウを借りてくることにします)。

まず，Fライン不当命令やAライン不当命令の例外が発行されたら，本来のファンクションコールへ飛ぶ前に，スタックフレームを除去して，68000と同じ構成にしてやります。そして，ファンクションコールの処理が終わった時点で，またスタックフレームを挿入してやり，rte命令でユーザモードに戻るという寸法です。

先ほどから「本来の処理へ飛ぶ前に」とか「本来のアドレス」などという言葉が出てきましたが，いったいどのように辻褄をあわせる作業を割り込ませるのでしょうか。

68000ではリセット時に0番地からPCを4番地からSPを読み込むという作業が行われていました。よって，この領域は電源投入時のことを考えれば，当然ROMでなくてはいけません。しかし，ソフトウェアによっては，残りの例外処理のベクタアドレスを書き換えたいこともあります。だから，「最初の2回のメモリアクセスはROMにして，それからあとはずっとRAMをあてがう」とか「RAMかROMかはバンク切り替えで決める」などという変則的アドレスデコーダを用意しなければなりませんでした。68020ではこの反省を踏まえて，例外処理のベクタテーブルのアドレスにオフセットがかけられるようになっています。

たとえば，このオフセットレジスタをかけるレジスタ（ベクタベースレジスタ）VBRに$ED0400を入れておけば，ベクタテーブルはX68000のSRAM内に設けることができます。そして，例外処理の中でもスタックを使った引数渡しを行わない処理(DMAの転送開始割り込みなど）はそのまま本来の処理ルーチンへ飛ばします。本来の処理ルーチンがどこにあるかは，0番地からの「本来の」ベクタテーブルを見て調べればわかります。FラインやAラインの例外が発生したときにだけ，スタックの「うわばみ」を書き換えて元の処理へ飛ばすのです。

そして，例外処理からユーザモードの処理へと復帰するとき，すなわちrte命令を発行する時点でスタックフレームを挿入してやります。

キャッシュの設定やVBRの設定を行うのは，SRAMのプログラム領域に書き込まれたプログラムです。HumanのSWITCHコマンドで，起動デバイス名にSRAMを指定すると，X68000は初期化に必要最小限の処理を終えると，ここに飛んでくるようにします。ROMのこのルーチンを眺めてみましたが，どうやら68020に引っ掛かるような命令はないようです。

幸いなことに，Human.sysやFSX.Xはディスアセンブラdisにかけると，ほぼ完全なかたちでソースリストが復元されます（もっとも，ラベル名などはメチャクチャだが)。試しに，Human.sysをdis.xにかけてみたのですがrte命令は12箇所しか使われていませんでした(バージョン2.03)。これなら，HumanやFSXの全容を知るのは不可能ですが，書き換えが必要なところを見つけるのは容易かもしれません。

＊　＊　＊

能書きは以上で終わりました。次回からは，いよいよ実機テストに入ります。速度はともかく，とにかく動くというバージョンです。その1号機が動いてからいろいろと速く動作させるための工夫を凝らしていきます。

図5-d　フォーマット$1のスタックフレーム

SP→ +00	ステータスレジスタ	
+02	プログラムカウンタ	
+04		
+06	0001	ベクタオフセット

図5-f　フォーマット$8のスタックフレーム

SP→ +00	ステータスレジスタ	
+02	プログラムカウンタ	
+04		
+06	0001	ベクタオフセット
+08	命令アドレス	
+0A		

図5-e　フォーマット$2のスタックフレーム

図5-g　フォーマット$9のスタックフレーム

SP→ +00	ステータスレジスタ	
+02	プログラムカウンタ	
+04		
+06	1001	ベクタオフセット
+08	命令アドレス	
+0A		
+0C	内部情報（4ワード）	
+0E		
+10		
+12		

参考文献

1）長澤克美，簡単に出来るGALライタの製作，トランジスタ技術91年7月号，CQ出版

2）長澤克美，GALライタのAバージョン対応法，トランジスタ技術91年11月号，CQ出版

3）CUPL 3.0 Manual

4）モトローラ，MC68020ユーザーズマニュアル

図5-h　フォーマット$Aのスタックフレーム

SP→ +00	ステータスレジスタ	
+02	プログラムカウンタ	
+04		
+06	1010	ベクタオフセット
+08	内部情報	
+0A	ステータスワード	
+0C	命令パイプステージC	
+0E	命令パイプステージB	
+10	データサイクルフォールトアドレス	
+12		
+14	内部情報	
+16		
+18	データ出力バッファ	
+1A		
+1C	内部情報	
+1E		

図5-i　フォーマット$Bのスタックフレーム

(で)のショートプロぱーてぃ――その41

音楽っていいな

Komura Satoshi　古村　聡

隙間風のため(で)氏は，風邪をひいてしまったようですね。それでも元気いっぱいの(で)氏がお贈りするのは，ショートでも音楽がカッコイイ「QREWD.BAS」と，半透明で表現力がアップする外部関数「CRTS.FNC」の2本です。

illustration：T.Takahashi

「しょうゆ」って漢字で書けますか？　私は「しょうが」って字すら書けません。こんばんわ，(で)です。

突然ですが，しょうが湯ってご存じですか？　先月の「風邪ひいた」って話の続きなんですが，あのあと鼻はナイアガラの滝のように流れまくり，頭は悪さをした孫悟空のように締めつけられ，熱は茶が沸かせるまでガンガン上がってしまい，「もうこのままでは死んでしまうっ！　ゼイゼイ」というところまでいってしまったのです。で，こりゃいかん，てんで薬屋に風邪薬を買いにいったのですが，ここのおばちゃんが親切な人で「体があったまるわよ」といって幅5センチ，長さ10センチ，いわゆる入浴剤にありがちな大きさの「しょうが湯」と書いた袋をオマケにつけてくれたんですねよね。

ふってみたら「さらさら」って粉の入ってる気配がする。こりゃあ，風呂に入れる粉末状の「登別」とか「草津」とかみたいな，いわゆる「日本の名湯シリーズ」に違いないってんでさっそく風呂に粉を入れて，つかってみたわけです。

んー，なんとも変わった香りがする(鼻が詰まってたんだけど)。おっかしいなー，なんなんだこれ，と思って，風呂からあがってよーく袋の裏を見てみたのです。

そしたら，なんと，袋に「150ccのお湯に溶かしてお飲みください」って書いてあるじゃあないですか！

「生姜湯［しょうがゆ］……熱湯におろししょうがと砂糖をまぜて飲む発汗剤。風邪気味のときによい。［小学館，日本語大事典より］」

飲み物だったんですねぇ，しょうが湯って。体も温まったからいいですけど……。

音楽はいいなぁ

んでは，まず，今月の1本目の作品にいきましょう。1本目のプログラムは神奈川県の笹井さんの作品でショートゲームQREWD.BASです。どーぞ！

QREWD.BAS for X68000
(要ジョイスティック，Z-MUSIC,MUSICZ.FNC，PCM8.XまたはMIDIボード&SC-55)
神奈川県　笹井進也

QREWDと書いて「くりゅーど」と読むこのゲームは，X-BASIC ver2.0以上用のショートプログラムです。ユーザーの環境によってMIDI版，PCM版を選ぶことができます。PCM版の場合には，フリーウェアPCM8.Xが必要です。で，このプログラムにはちょっとした下準備が必要です。

まず，リストを打ち込む前に，お好みのPCMファイルを3つ用意してください。こいつらがミス，ゲームオーバー，面クリアの効果音になります。ZVT.Xなどを使い自分でサンプリングするのがベストですが，Z-MUSICのシステムに入っているPCMファイルで代用してもかまいません。

どちらにしても，できるだけセンスのよいものを使いましょうね(ま，思いっきりミスマッチでヘンな音ってのもシュールでいいかもしれないけど)。これらをプログラムと同じディレクトリにそれぞれ，“QRD1.PCM”“QRD2.PCM”“QRD3.PCM”というファイル名にしておきます。

また，PCM版ではZ-MUSICシステムのディスク2に入っている“WDS1.PCM”“WDK1.PCM”“CH1.PCM”の3つも同じのディレクトリにコピーしてください。

で，あとはリストを打ち込んで(PCM8版はリスト1をそのまま，MIDI版は770行までリスト1を打ち込み，780行以降はリスト2を入力してください)から，

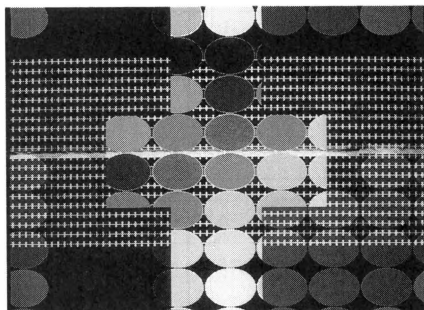
```
A>PCM8(←MIDI版は不要)
A>ZMUSIC -U -P128 -S68SND.ZMS
A>BASIC
```

とZ-MUSICを組み込んだ状態でBASICを立ち上げ，プログラムをRUNします。

で，ゲームの遊び方です。タイトルが出たら，ジョイスティックのトリガを押すことでゲームがスタートします。下から浮いてくるドットをジョイスティックで左右にコントロールして，壁にぶつからないように最上部のゴールへ導いてください。ドットの動きは，ジョイスティックを倒した方向に加速度がかかります。ジョイスティックをニュートラルにしている状態では，現在進んでいる方向に等速直線運動をするわけです。

上方向への速度は常に一定です。面が進むに従って壁の間隔が狭くなり，難しくなりますが，その分左右の加速度も大きくなります。移動量は増えますが，コントロールは難しくなります。

クリアするとラウンド数×100の点数が入りますが，それ以外に芸術点も入ります。つまり芸術的にゴールに入ればそれだけ，

QREWD.BAS

CRTS.FNC

高得点が得られるというわけです(どう入ると芸術的なのかは内緒ですけど……)。3回ミスするとゲームオーバーです。

それにしても久しぶりのゲームプログラムでありますねぇ。お兄さんは嬉しいぞ。すりすり。

ゲーム内容はショートプロにありがちな慣性ドットものなんでありますが，ゲームはすっごくかっこいいです。そう，Z-MUSIC&PCM8(またはSC-55)を使ってゲーム中の音楽と効果音を出しているのですが，このゲームミュージックがよいのですよね，超カッコイイの。そうか，ショートでもかっくいい音楽をつけるとゲームもすごくかっこよく見えてしまうのだなぁ。

実をいうと私は，あんまり音楽には自信のない人(かなり控え目な表現である。人は私を史上最大の爆発・カラオケ騒音野郎と呼ぶ)なので，自分の作ったゲームにミュージックをつけたことがなかったのですが，これ見てしまうとなんだか，自分の作ったものにも音つけたくなってしまいますね。でも，作曲なんてぜんぜんできないしなあ。はぁ。私といたしましては笹井さんがうらやましいかぎりです，はい。

優良サンプルプログラム

んでは，続きまして今月の2本目のプログラムは，広島県の清水さんの作品で半透明&プライオリティをBASICから使う，CRTS.FNCです。どうぞ～。

CRTS.FNC for X68000

(要Cコンパイラ)

広島県　清水弘和

このプログラムはBASICの外部関数でありながらほとんどCで書かれています。で，ちょっとリストがコマ切れになってしまっているんですが，CRT.S，PR.C，TR.Cという名前でリスト3，4，5を入力してください。

リストはコマ切れですけどコンパイルは一発でできます。打ち間違いがないことを確認したら，

```
A>CC /Y /FxCRTS.FNC CRTS.S TR.C PR.C
```

としてください。これでOKです。そして，コンパイルが終わったら，BASIC.CNFファイルに，

```
FUNC=CRTS
```

と書き加えておいてください。このプログラムはX-BASICに3つの関数を拡張してくれます。で，その増える関数ですが，まずひとつは画面間のプライオリティを変更させるための関数で，prio(r)です。引き数はint型で戻り値はありません。

rは画面間のプライオリティ，つまり，優先順位を示していて，0～5の範囲で使えます。スプライトをS，テキストをT，グラフィックをG，左にあるほうが優先順位が高いとすると，表1のように順位を変えることができます。

2つめはグラフィックのプライオリティを変えるgprio(r) です。rでグラフィック画面のページを決定するもので，引数によって表2のようになります。

最後は半透明機能。

trparent(r)という名前で，引数rは1～11の間で使うことができます。

このtrparentはグラフィックの最もプライオリティの高いページといろいろなページとの半透明機能を使えるようにします。その内容は，rを1から11まで変えることでそれぞれ，表3のように変わります(セカンドページと書いてあるのは，グラフィックの2番目にプライオリティの高いページのことです)。特殊プライオリティは，グラフィックのプライオリティがテキストやスプライトより低いとき，グラフィックの最もプライオリティの高いページのプライオリティを，テキストやスプライトよりも上にするものです。

9～11は，r=5に加えてビデオコントローラのレジスタR2の最上位ビットを立て

表1　優先順位の一覧

r	
r=0	STG
r=1	SGT
r=2	TSG
r=3	TGS
r=4	GST
r=5	GTS

表2　ページ間のプライオリティ

r	高 ←→ 低			
0	0	1	2	3
1	0	1	3	2
2	0	2	1	3
3	0	2	3	1
4	0	3	1	2
5	0	3	2	1
6	1	0	2	3
7	1	0	3	2
8	1	2	0	3
9	1	2	3	0
10	1	3	0	2
11	1	3	2	0
12	2	0	1	3
13	2	0	3	1
14	2	1	0	3
15	2	1	3	0
16	2	3	0	1
17	2	3	1	0
18	3	0	1	2
19	3	0	2	1
20	3	1	0	2
21	3	1	2	0
22	3	2	0	1
23	3	2	1	0

表3　半透明機能

r	
r=0	戻す
1	テキストのパレット0の色
2	テキスト，スプライト
3	セカンドページ
4	(r=2)+セカンドページ
5	(r=2)+テレビ/ビデオ
6	(r=3)+テレビ/ビデオ
7	(r=4)+テレビ/ビデオ
8	特殊プライオリティ
9	5と同じ(R2最上位ビットON)
10	6と同じ
11	7と同じ

動かないよ，と思う前に(4)

今月のQREWD.BASでは，Z-MUSICとPCM8を使いますね。今月はこのような常駐ソフトを使うときの注意点についてです。

★PCM8.XとPCMDRV.SYSを登録してますか？

今回のQREWD.BASはMIDI版はZ-MUSICが，PCM8版ではZ-MUSICとPCM8.Xが必要になります。必ず常駐させてからリストを実行させてください。PCM8.Xは1992年6月号付録の「創刊10周年記念PRO-68K」に入っています。また，PCM8.Xが常駐している場合でもCONFIG.SYSにPCM-DRV.SYSの登録も必要です。CONFIG.SYSももう一度確認してみてくださいね。

★オプションを間違えていませんか？

それから，今回のZ-MUSICのようにほかのソフトを事前に常駐させる場合，オプションの間違いに気をつけてください。今回の場合は，-Uオプションをつけ忘れると，リストは正しくても「規定外トラックエラー」になります(自分でやって大騒ぎした)。気をつけましょう。

★ほかの常駐ソフトは大丈夫？

ごくまれにソフトによっては指定されている以外の常駐ソフトなどのせいで動かなくなっている可能性もあります。普通，常駐ソフトというのは一定のマナーに沿って作られるのですが，たまにお行儀の悪いソフトが存在するのです。どうしても動かない場合には，怪しそうな常駐ソフトを外し，リトライしてみてください。

★どうしても動かない場合……

で，本当にバグがあるのを確認した場合にはOh!Xのバグ電話に電話してください。そのときには，できるだけエラーが起きた状況を細かく説明してもらうことになる(エラーが出た行，使っているソフトのバージョン，環境)ので，メモをとってからしていただけると非常にありがたいです。電話しながらキーボードを叩くなんてことがないようにね。

ることで，スーパーインポーズ時でもビデオ画像を表示しないようにしています。

ところで，テレビ/ビデオとなっているところはカラーイメージユニット用ですね。

半透明や特殊プライオリティの領域指定は，プライオリティの最も高いグラフィックのパレットコードの最下位ビットを1にして（偶数にして）指定します。

……ということですが，おわかりいただけましたでしょうか。ちょっと文章で説明すると難しいようだけど，サンプル（リスト6）を実行すれば，ひと目でどんなことをしているかわかると思いますよ(それでもわからない場合は，InsideX68000を読むのがオススメかな)。

んー，それにしても，よく考えてみたらC言語で書かれたX-BASICの拡張関数ってショートプロ初，かな。やっぱりこの手のプログラムはCで書くのが楽ですよねぇ，ソースコードデバッガもあるし(このプログラムの場合は，どっちのデバッガでもあんまり手間は変わらんかな？　ほとんどアセンブラみたいなプログラムだし……)。

これを参考にすればCしか使えない人でも，".s"の部分だけそのままパクッてしまえば(さすがにBASICから呼び出されるときの関数名とかは変えるだろうけど)，とっても簡単にX-BASICの関数を作ることができると思います。それにこのプログラムは，関数で使われている特殊プライオリティなども結構実用になるサンプルとして使えそうだし，これから自分でプログラムを作りたい！　という人にはいいお手本のエッセンスの詰まったプログラムですよね。大変よくできました。

さぁて，風邪も治ったことだし，なんだか出かけたくなってきたなぁ。温泉にでも行こうかな，しょうがの入ってない……(だから自分で蒸し返すなっての。恥ずかしいんだから)。んではまた来月。

リスト1　QREWD.BAS(PCM8版)

```
 10 /**********************************************
 20 /**    QREWD68k for PCM8    **
 30 /**                                            **
 40 /** .programed by SHINYA SASAI  **
 50 /**********************************************
 60 int i,j,iy,ro,rn,x,y,yy,sc,hs,qrd , spl,pc ,ap
 70 float w,ww,kk
 80 str a[255],b[255]
 90 randomize(len(mid$(time$,4,2)+right$(time$,2)))
100 mu() : title() : while 1
110 apage(0) : cls :m_play(1,2,3,8,9,10)
120 sc=0:qrd=3:ro=1:iy=16:r=100:w=0:ww=0.2#:pc=1
130 /* GAMEN-----------
140 while qrd>0
150 fill(0,0,511,511,0)
160 box(0,16,511,511,6)
170 for i=1 to 480/r-1
180 line(0,iy+i*r,511,iy+i*r,6)
190 rn=rnd()*410+1
200 line(rn,iy+i*r,rn+100,iy+i*r,0)
210 next
220 rn=rnd()*410+1
230 fill(rn,16,rn+100,18,57)
240 x=256 : w=0
250 color 7 : locate 0,0:print"SCORE:";sc
260 locate 18,0:print" HIGH:";hs
270 locate 36,0:print"ROUND:";ro
280 locate 52,0:print"LEFT:";qrd-1  : color 3
290 locate 19,14:color 7
300 print"ROUND ";ro;" READY"
310 locate 20,17:print"PUSH TRIGGER"
320 while strig(1)=1 : endwhile
330 while strig(1)=0 : endwhile
340 locate 19,14:print space$(30)
350 locate 20,17:print space$(30)
360 /* MAIN=======
370 for yy=1 to 492 : y=510-yy
380 w=w+(stick(1)=4)-(stick(1)=6)
390 xx=x:x=x+w*ww
400 if x<1 or x>510 or point(x,y)=6 then break
410 pset(x,y,255):pset(xx,y+1,0)
420 next
430 if point(x,y-1)=57 then nround() else dead()
440 endwhile
450 symbol(52,180,"GAME OVER",2,3,2,235,0)
460 locate 20,20:color 7:print"PUSH TRIGGER"
470 repeat : until strig(1)=1
480 endwhile
490 end
500 func dead() :/* BAKUHATSU---
510 if qrd>1 then m_play(12) else m_stop()
520 m_play(13)
530 qrd=qrd-1
540 for i=0 to 100
550 line(x+rnd()*200-100,y+rnd()*200-100,x,y,rnd()*256)
560 next
570 endfunc
580 func nround():/* CLEAR!!-----
590 m_play(14)  : if ro>7 then ww=ww*1.05# else ww=ww*1.25#
600 ww=int(ww*100)/100 : ap=abs(w*ww)*100
610 sc=sc+ro*100+ap : if sc>hs then hs=sc
620 locate 14,13:print"ARTISTIC POINTS:";ap
630 for i=0 to 5000 : next : locate 14,13:print space$(42)
640 ro=ro+1
650 if ro>7 then r=r-2 else if ro>3 then r=r-4 else r=r-10
660 endfunc
670 /* TITLE------
680 func title()
690 screen 1,2,1,1:console ,,0:vpage(0)
700 tenten() : apage(0)
710 for i=0 to 7 : for j=0 to 7
720 symbol(20+j,70+i,"QREWD",12,8,1,58,0)
730 next:next
740 for i=0 to 3 : for j=0 to 3
750 symbol(22+j,72+i,"QREWD",12,8,1,0,0)
760 next:next
770 symbol(360,200,"68K",3,3,2,36,0)
780 vpage(3)
790 locate 20,20:color 7
800 print"PUSH TRIGGER"
810 locate 14,28
820 print"COPYRIGHT BY PUMPKIN SOFT 1985,1992"
830 repeat : until strig(1)=1
840 cls : endfunc
850 func tenten()
860 apage(1):for i=0 to 400
870 pset(rnd()*512,rnd()*512,rnd()*32*8)
880 next : endfunc
890 endfunc
900 /* MUSIC==========
910 func mu()
920 m_pcmset(36,"WDK1.PCM"):m_pcmset(38,"WDSD1.PCM")
930 m_pcmset(40,"CH1.PCM")
940 m_pcmset(60,"QRD5.PCM"):m_pcmset(62,"QRD6.PCM")
950 m_pcmset(64,"QRD4.PCM")
960 m_init() : m_ch("fm")
970 m_alloc(1,500):m_alloc(2,500):m_alloc(3,500)
980 m_assign(1,1):m_assign(2,2):m_assign(3,3)
990 for i=8 to 14:m_alloc(i,100):m_assign(9,i):next
1000 a="[do]L16 |:4 a8aa<e>aa<e>:||: g8gg<d>gg<d>:| f8ff<c>fc<c
> g8gg<d>gg<d>
1010 b="|:4 a8aa<e>aa<e>:||: g8gg<d>gg<d>:| a8aaggbg a8aaa8gg [
loop]
1020 m_trk(1,"v13 o2 @10 ") : m_trk(1,a) : m_trk(1,b)
1030 a="[do]L16 c>bab<c2>b4 <c>bab<c2d8c8 >bagab2a8b8 <c2d4>b4"
1040 b="<c>bab<c2>b4 <c>bab<c2d8c8 >bagab2a8g8 a1[loop]"
1050 m_trk(2,"v15 o5 @19 ") : m_trk(2,a) : m_trk(2,b)
1060 a="[do]L1 |:'ea<c'384 'dgb'| 'c2fa' 'd2gb':| 'ea<c' [loop]
1070 m_trk(3,"v13 o4 @25 ") :m_trk(3,a)
1080 m_trk(8,"o2 [do]|:32 e8e16e16 :|[loop]")
1090 m_trk(9,"[do]L4o2|: |:3 crcr :|crcr8 c8 :|[loop]")
1100 m_trk(10,"[do]L4o2|: |:3 rdrd :|rdrd8.d16[loop]")
1110 m_trk(12,"o4c*384") : m_trk(13,"o4d*384") : m_trk(14,"o4e*
384")
1120 m_tempo(136)
1130 endfunc
```

リスト2　QREWD.BAS(MIDI版)

```
 780 /* MUSIC===
 790 func mu()
 800 m_pcmset(60,"QRD2.PCM"):m_pcmset(62,"QRD3.PCM"):m_pcmset(6
4,"QRD1.PCM")
 810 m_init() : m_ch("midi")
 820 dim char rs(3)={&H40,0,&H7F,0}
 830 m_roland(&H10,&H42,rs)  /*SC55を工場出荷状態に初期化
 840 dim char vr(15)={1,1,6,1, 0,0,0,0 ,0,5,0,0 ,0,0,0,0}
 850 sc55_v_reserve(vr)          /*SC55ボイスリザーブ
 860 for i=1 to 10:m_alloc(i,200):m_assign(i,i):next
 870 for i=12 to 14:m_alloc(i,20):m_assign(25,i):next
 880 for i=1 to 10 : m_trk(i,"@i$41,$10,$42") : next
 890 a="L16 [do]|:8 c8<cc> :||:8 >b-8<b-b- :||:8 >a8<aa :||:8 >
g8<gg  :| [loop]
```

```
 900 m_trk(1,"n1 v10 @p76 o2 @35 @e80,40") : m_trk(1,a)
 910 a="L4[do]|:3 e.d.fe.c2r8  :| d2c2>b<cdf [loop]
 920 m_trk(2,"n2 v10 @p52 o5 @66 @m30 @h36 @e120,40") : m_trk(2
,a)
 930 a="[do] 'eg<c'384 'egb-'384 'cea'384 '>b<dg'384 [loop]
 940 m_trk(3,"n3 v8 o4 @52 @e40,40") : m_trk(3,a)
 950 m trk(8,"n10 o2@u100 L16[do]|:64 g-g- :|[loop]")
```

リスト3 CRT.S

```
 1: *        CRTS.s  *
 2: *cc CRTS.s PR.c TR.c /Y /FxCRTS.FNC
 3:          .include        fdef.h
 4:          .xdef           _main,err_msg
 5:          .text
 6: **INFORMATION TABLE**
 7: _main:
 8:          .dc.l   No,No,No,No,No,No,0,0
 9:          .dc.l   Token_table,Param_table,Execution
10:          .dcb.l  20,0
11: No:
12:          rts
13: Token_table:
14:          .dc.b   'prio',0        *PRIORITY
15:          .dc.b   'gprio',0       *GRAPHIC PRIO.
16:          .dc.b   'trparent',0    *TRANSPARENT
17:          .dc.b   0
18:          .even
19: Param_table:
20:          .dc.l   param,param,param
21: param:
22:          .dc.w   int_val,void_ret
23: Execution:
24:          .dc.l   _prio,_gprio,_trparent
25:          .data
26:          .even
27: err_msg:
28:          .dc.b   '引数が無効です。',0
29:          .even
```

リスト4 PR.C

```
 1: /*
 2:          TRANSPARENT
 3: */
 4: #include <iocslib.h>
 5: int trparent(int param,int prm0,int r)
 6: {
 7:          char *p=0xE82600;
 8:          char d[12]={ 0,92,29, 30, 31, 61,
 9:                      62,63,20,189,190,191};
10:          if( (r<0) || (r>11) ){
11:              #asm
12:                  lea.l   err_msg,a1
13:              #endasm
14:              return(1);}
15:          B_BPOKE(p,d[r]);
16:          return(0);
17: }
```

```
 960 m_trk(9,"n10 L2o2@u110 [do]|:8 c c  :|[loop]")
 970 m_trk(10,"n10 v14 L4o2@u115 [do]|:8rdrd  :|[loop]")
 980 m_trk(12,"o4c*384") : m_trk(13,"o4d*384") : m_trk(14,"o4e*
384")
 990 m_tempo(140)
1000 endfunc
```

リスト5 TR.C

```
 1: /*
 2:          PRIORITY
 3: */
 4: #include <iocslib.h>
 5: int prio(int param,int prm0,int r)
 6: {
 7:          char *p=0xE82500,d[6]={6,9,18,33,24,36};
 8:          if( (r<0) || (r>5) ){
 9:              #asm
10:                  lea.l   err_msg,a1
11:              #endasm
12:              return(1);}
13:          B_BPOKE(p,d[r]);
14:          return(0);
15: }
16: int gprio(int param,int prm0,int r)
17: {
18:          char *p=0xE82501;
19:          char d[24]={228,180,216,120,156,108,225,177,
20:                      201, 57,141, 45,210,114,198, 54,
21:                       78, 30,147, 99,135, 39, 75, 27};
22:          if( (r<0) || (r>23) ){
23:              #asm
24:                  lea.l   err_msg,a1
25:              #endasm
26:              return(1);}
27:          B_BPOKE(p,d[r]);
28:          return(0);
29: }
```

リスト6 SAMPLE.BAS

```
 10     str s="++++++++" : char p(255) : int xc,yc
 20 screen 1,2,1,1:console ,,0:color 5:sp_init():vpage(0)
 30 for i=0 to 255: p(i)=i/17            :next  :sp_def(0,p,1)
 40 for i=0 to 31 : sp_move(i,i*16,0,0):next
 50 locate 0,5 :for i=0 to 159:print s;:next: apage(1)
 60 for y=0 to 7 : for x=0 to 7
 70   xc=x*64+32:yc=y*64+32
 80   circle(xc,yc,31,x*y*3+100)  :  paint(xc,yc,x*y*3+80)
 90 next         :next
100 apage(0)                :fill(  0, 0,511,511,201)
110 fill(120,180,391,331,0):fill(200, 0,311,511,  0)
120 prio(5)  :  trparent(4)  :  sp_disp(1)  :  vpage(3)
130 while 1:for k=0 to 1:for i=0 to 124
140   home(1,i*8+504*(i>62),i*8+504*(i>62)):for j=0 to 31
150   sp_move(j,j*16,-496*(k=1)+i*4+8*(k=1)*i,0):next
160 next   :next         :endwhile
```

ぱーてぃハンズ(4)

さあて，今月は「2人で殴りあう」前に，ひとりで動かすときのこまごまとした部分を直していきましょう。んーと，まず，このプログラムのキャラクターは，いつも同じ方向を向いているのですよね。でもこれじゃぁ，2人で向き合って殴れないですよねぇ。2人とも右向いちゃってんだもん。ってことで，キャラクターに振り向いてもらうことにしましょう。では，スタート。

あんどおあ〜といくすおあ

で，プログラムの説明をする前に，ちょっと聞きます。論理演算って知ってますか？　今回のプログラムで使う関数では論理演算を少し使うのでその説明をしておきましょう。知ってる人は，次の見出しに飛んでくださいね。

さて，論理演算というのは，AND，OR，NOT，XORという演算子(加算や減算みたいなもんですね)で行う，いっぷう変わった計算方法なんです。加算だったら，

0+0=0
1+0=1
0+1=1
1+1=2

となるんですが，ORという演算は，

0 OR 0=0
1 OR 0=1
0 OR 1=1
1 OR 1=1

という結果になるんですね。なんとなくわかるでしょう(なんとなく変かな？)。主な論理演算についての結果を表1に載せておきます。

X-BASICでは，この論理演算子を加算や減算と同じように使うことができます。たとえば，

PRINT 130 XOR 3

とすると130と3のXORの結果である，

129

という表示をしてくれるんです。

ま，とにかく論理演算といって，表1みたいな計算のしかたがあって，それはX-BASICでふつうに加算や減算と同じように使える，ということだけを覚えておいてください。わかったかな？

反転するよ

さ〜て，先に飛んできてしまった人はお待たせしました。

いまさらゆうのもなんですけど，X68000のスプライトって，どんなことができてどんなこと

ができないか覚えてますか？　1つひとつの絵をスプライトとして出すほかに背景用のバックグラウンドとして使えるとか，いろいろありましたね。

んで，その機能のなかに「スプライトの上下左右反転表示」っていうのがあるんです。これはどういう機能かというと，その名のとおり「スプライトを表示するときに上下や左右をひっくり返して表示してくれる」機能なんですよね。ここでは，キャラクターを反転して歩かせたいんですから，こいつをそのまま使ってしまえばいいわけだ。

この左右反転機能は，BASICからだとsp_set()関数で使うことができます。sp_set()関数はマニュアルによると，

●sp_set(s, [x], [y], [pd] [pr])

pd……パターンデータ

pdのビット

ビット15……垂直反転(0：通常，1：垂直反転)

ビット14……水平反転(0：通常，1：水平反転)

ビット8～11……パレットブロック

ビット0～7……パターンコードcd

となっています。

つまり，前回まで使っていたsp_moveのパターンコードcdのビット14を1にしてやれば，左右がひっくり返ってくれるというわけです。この場合なら2^{14}＝16384をスプライトのパターンコードと，ORしてsp_set()関数で定義してやればスプライトの左右が見事にひっくり返ってくれる，というわけですね。

先月のリストがある人は，そのsp_move()をsp_set()に変えて，スプライトコードを16384とORして試してみてくださいね。

XORよ，あなたは偉かった

で，先月のリストで試してみた人はわかると思うんですが，これだけではちょっとまずいんですよね。というのもたしかにスプライトのパターン自体は左右反転してくれるんですが，キャラクター自体は逆になってくれません。なぜなら，パターン自体は左右反転してるんですけど，並び方が正しくないんです。

そう，キャラクターは6個のスプライトが並んでいるんですから，この並びも左右逆にしなければいけないんです。パターンが逆だったときに，どうやって並びを逆にすればいいんでしょう。

ここで，パターン反転をするときには，スプライト表示ルーチンに関数に1，しないときには，0がメインのルーチンから渡されるとします。先月号のsp_move()を使ったときのスプライトパターンの引数は，

pnum＋i＋iy×8＋256

となっていましたから，今回はsp-set()を使うときに，

(pnum＋i＋iy×8＋256)OR(16384×d)

としています。こうするとdに1が渡されたとき，16384×1＝16384とのORがとれますし，0のときには，16384×0＝0とのORがとれます。表1を見ればわかりますけど，0とのORをとる，つまり，もう一方の数値がそのまま答えになるっていうことです。

で，このスプライトの並びのほうなんですが，ここでスプライトを並べるときに，先月号のリストでこんなことをしているのに気がつきませんか？

```
for ix＝0 to 1
   ⋮
sp_move(……
   ⋮
   ⋮
next
```

よ～くリストを読んでみるとわかるんですが，このixが0のときに6個のスプライトのうちの左半分3個を，1のときに右半分の3個を描いているのですよね。するってぇと，sp-move()に渡してX座標の値のなかでixを使ってるんだから，このixが0のとき1を，1のとき0を渡してやれば，パターンの並び自体も左右反転できるんですよね。だから，……ixが0でdが0のときは0でしょ。ixが0でdが1だったときは1で……うん，こういう感じになるように値を渡してやればいいんですよね。

ix	d	引数
0	0	0
1	0	1
0	1	1
1	1	0

これってどこかで見てませんか？　そう，さっきの論理演算の“XOR”の表とまるっきり同じなんですよぉ。おお，なんたる偶然。

とすると，先月のリストのPutSprite()関数で，sp_move()のX座標の式でixだったところを，ixとdのXORにしてやればいいんですね。つまり，

x＋ix×16

だったのを，

x＋(ix XOR d)×16

にしてしまうだけでいいんですねぇ。ううむ。論理演算は偉いっ！

というわけで今月の成果

という方針で直してやったのが，今月のリストです。ジョイスティックが左に押されたりしたときに，キャラクターを逆に向かせることができるように，スプライトを表示する部分であるPutSprite()関数にd＝1を渡しています。

実際プログラムをRUNしてみると，ちゃーんと，キャラクターが反転してくれるのがわかるでしょう。このページってモタモタ，いきあたりばったりに進んでいるわりには，ちゃんとプログラムはできていってる(ような気がする)んですよねぇ。うーん，偉い。

ところで，先月，えええっと，パンチやキックしながら歩いてしまう，という点を直そうという話をしたのですが，今月のリストで一気に直してしまいました。それほど複雑なことをしたわけではないのですけど，とりあえず，来月は，その改良点のあたりのお話とうまくいったら，いよいよキャラクターを2人に増やしてしまいましょう。

うう，でもそんなにうまくいくかなぁ。ちょっと不安を残しながら来月に続きます。

表1　論理演算一覧表

OR		
0	0	0
0	1	1
1	0	1
1	1	1

AND		
0	0	0
0	1	0
1	0	0
1	1	1

XOR		
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
0	1
1	0

リスト

```
 1000 screen 0,0,0,0
 1010 int x,y,h,dx,k,l
 1020 x=48:y=128:dh=-4:l=0:d=0
 1030 sp_disp(1)
 1040 sp_on(0,1)
 1050 PutSprite(0,x,y,0)
 1060 if(h = 0 and k=0 and l=0) then{
 1065 switch (stick(1))
 1070      case 4: walk(x,y):x=x-4:d=1:break
 1080      case 6: walk(x,y):x=x+4:d=0:break
 1090      case 7: h=-8:dh=-8:dx=-4:d=1:break
 1100      case 9: h=-8:dh=-8:dx=+4:d=0:break
 1110      case 8: h=-8:dh=-8:dx=0:break
 1115 endswitch
 1120 if(strig(1) = 1) then PutSprite(2,x+4,y,d):k=2:l=1
 1130 if(strig(1) = 2) then PutSprite(4,x+4,y,d):k=4:l=1
 1150 }else{
 1160 if(strig(1) = 1 and h = 0 and k=0 and l=1) then PutSprite(
0,x,y,d)
 1170 if(strig(1) = 2 and h = 0 and k=0 and l=1) then PutSprite(
0,x,y,d)
 1180 }
 1185 if(strig(1) = 0 and h = 0 and k=0) then PutSprite(0,x,y,d)
:l=0
 1190 if k<>0 then k=k-1
 1200 if(h<>0) then x=x+dx
 1210 if(x<16) then { dx=dx*-1:x=x+dx:x=16 }
 1220 if(x>224) then { dx=dx*-1:x=x+dx:x=224 }
 1230 if(h<>0) then h=h+dh:x=x+dx:PutSprite(6,x,y+h,d):if(h<-64)
then dh=8
 1240 if(h<>0) then h=h+dh:x=x+dx:PutSprite(6,x,y+h,d):if(h<-64)
then dh=8
 1250 goto 1060
 1260 func walk(x, y)
 1270    int ix,iy
 1280    for  ix=0 to 1
 1290       iy=2
 1300       sp_set(ix+iy*8,x+ix*16,y+iy*16,8 +ix+iy*8 + 256)
 1310    next
 1320 endfunc
 1330 func PutSprite(pnum, x, y, d)
 1340 ·  int ix,iy
 1350    for  ix=0 to 1
 1360       for iy=0 to 2
 1370 sp_set(ix+iy*8,x+(ix xor d)*16,y+iy*16,pnum+ix+iy*8 + 256
+16384*d)
 1380       next
 1390    next
 1400 endfunc
```

第129部　BLACK JACK

●BLACK JACK

その昔，マイコンという名が全盛の時代。キャラクタグラフィックで作られたゲームとして，このBLACK JACKがよく登場したものです。

ルールの単純さもあって，ショートショートプログラムの題材，BASIC講座でのサンプルプログラムなど，ずいぶんとお目にかかったような気がします。

また，当時，かなりお約束的なゲームとして，ポーカーゲームもありました。こちらも，いまだS-OSでは発表されたことはありません。ただし，今回のBLACK JACKが基本セットですから，ポーカーゲームのほうは，ひとひねりもふたひねりもほしいですね。ルール自体は変更のしようもないかもしれませんが，遊んでいて面白ければなんでもありでしょう。誰か挑戦してみませんか？

●コアシステム

さて，こういったカードゲームが発表されるたびに思うことなのですが，カードキャラクタなどを簡単に扱えるライブラリ，もしくはコアシステムがほしくなります。

確かに，システム化することによってデザインの自由度は減ります。しかし，そのシステムを使うことによって，アプリケーションの作成の手間がかなり軽減されるはずです。カードのデザインだったら，ある程度固定化しても問題はなさそうだし，利用価値は結構ありそうです。

などと，いわくありげないい方をしていますが，これは皆さんもご存じのとおり，X68000用に発表されたCARDDRV.Xによって実証済みです。はっきりいって，いままでOh!Xで発表されたどのシステムよりも(S-OSは除きましょう）アプリケーションが発表されて，現在も使われ続けています。

これはひとえに，ある特定分野，この場合はカードゲームの中核を簡単に記述できること，そしてカードゲームというバリエーションの多い分野で使われるものである，ということがあげられるでしょう。

以前，ウィンドウシステムまで発表されたことのあるS-OSですから，それほど実現不可能な話ではありません。逆に，キャラグラであることの有利さをふんだんに使った，面白いものができそうです。特定の大きさのカードをプットするだけでなく，さまざまなアニメーション機能（回転，カードをめくる動作，拡大縮小）などの付加機能があれば完璧でしょう。

と，今月はお願いごとばかりいっているような気がしますが，試みとしては面白いと思いませんか？

●S-OSの系譜(41)

1990年３月号では,「史上最強のアブソリュートアセンブラ」とうたわれたOHM-Z80が発表されました。制作者は,SLANGなど大作を発表し続けていた，かなりのヘビーユーザーである大貫氏です。

なにが史上最強かというと，疑似マクロ命令によるZ80命令の拡張,高級言語でお馴染みの構造化制御文が記述可能であったり，もはやアセンブリ言語とは呼べない！　とまでいわしめるほどのものでした。

大貫氏は,「アセンブリ言語に近い高級言語があるなら，高級言語に近いアセンブリ言語があってもいいじゃないか」という発想をもとに，このOHM-Z80を制作したそうです。

そういった経緯で制作されたOHM-Z80の疑似マクロ命令は,本当に強力でZ80というCPUは,直交性に優れたCPUであると錯覚するほどのものでした。特に，転送命令のLD命令など，すべてのレジスタ間で転送が可能だし，メモリ間の転送まで行えるという充実ぶり。ぎっちり埋まった命令表を見たときは，感動すら覚えました。

ただ，アセンブラとはいえないほどの多機能さがあだになり，掲載には多少の時間がかかったのも事実です。なぜなら，アセンブラの魅力のひとつである，シンプルさが損なわれる危険性があったためです。

アセンブラを使うことは，直接CPUと対話するという意味合いをもっているといえ，疑似マクロ命令によって本来の命令が覆い隠されることは，問題があるんじゃないか。という，親アセンブリ言語ユーザーからの警告もありました。

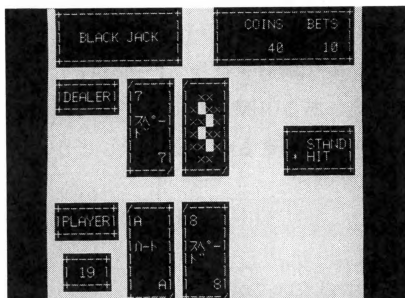
また，わかって使うならともかく，アセンブラの初心者が触れるのは控えたほうがいいかも……とまでいわれたのです。

シンプルな構造を目指したREDAとは，対極の立場にあるOHM-Z80ですが，この志の高さは見習うべきものがあるでしょう。

1993■インデックス

BLACK JACK
(要SLANG)

Watanabe Keiichi
渡辺 慶一

今月は古典的なカードゲームである，BLACK JACKをお届けしましょう。役によってボーナスがつくなど，ルールはゲームセンターにあるものに近い仕様です。目指せ，ブラックジャック。

いきなりですが，私はPC-9801ユーザーです。以前からシリーズ全機種共通システムS-OS "SWORD" は素晴らしい，と思っていましたがパソコンはPC-9801しか持っていなかったので，ただ指をくわえて見ているしかありませんでした。

しかし，1990年6月号のPC-286用S-OS "SWORD" によって，PC-9801でもS-OSを動かすことができるようになりました。制作者の遠藤さん（そしてX68000版の宮島さん）は本当にすごい人でしょう。これであとは，FM TOWNSとDOS/Vマシンで動くようになるとほぼ世界制覇？　ですね。

さて，PC-286用S-OS"SWORD"でS-OSができるぞ！　と思ってもどんなソフトを作るか迷いました。システム関係は，私の手に負えないし，Z80のマシン語がわからないのでアクションゲーム関係は無理そうだし……そこで，ふと思いついた簡単そうでまだ発表のない「BLACK JACK」を制作してみました。

例によってグラフィックキャラクタのないS-OSですから，カードのデザインが気に入らない人は各自変更したほうがいいかもしれません。

ルールについて

実は，私はブラックジャックについての正確なルールを知らないので，ゲームセンターによくあるコインゲーム機を参考に，自分でアレンジしてしまいました。ですから，世間一般のルールとは違うところがあるかもしれません。

それでは，この「BLACK JACK」で使っているルールを説明していきましょう。まず，プレイヤーは2人，ディーラー（コンピュータ）とプレイヤーです。ディーラーとプレイヤーにそれぞれカードが配られ，カードの合計が21に近いほうが勝ちとなります。ただし，21を超えてしまうと無条件に負けとなります。カードはジョーカーを除いた52枚を使用します。

始めに賭け金を決めます。ベットは最大10枚までできます。掛け金を決めたら，リターンまたはスペースでスタートです。

まず，1枚ずつ，プレイヤーに2枚，ディーラー（コンピュータ）に2枚（1枚はふせてある）配ります。自分の手札が，21になるべく近くなるようにしなければなりません。数のかぞえ方は，A（エース）は1または11のどちらかで，10以上はすべて10とかぞえます。

もうすでに自分の手札が21に十分近いと思うなら "STAND" にします。もう1枚ほしいなら "HIT" にして1枚もらいます。"HIT" は何回もできますが，画面の都合上5回（合計7枚）しかできません。まあ，7枚もあって，合計が21以下ということはほとんどないので問題ないと思います。もし，21を超えてしまったらそこで負けです。

次に，ディーラーが同じことをして勝負を決めます。プレイヤーは役がつくと，ディーラーに数で負けていても勝つことができます。プレイヤー，ディーラーともにブラックジャック（2枚で21になる手札）は最強で，相手もブラックジャックでないかぎり勝てます。なお，当然のことながらイカサマはいっさいしていません（ゲームセンターのは絶対していると思う）。

プレイヤーが勝つか引き分けた場合，今の配当金をそっくりそのまま次のゲームにかける "CONTINUE" か，手持ちのコインに組み入れる "TAKE SCORE" を選択してください。

持ち金が0になるとゲームオーバーになります。詳しくは，ゲームをやってみるのがいちばんでしょう。

操作&役について

モードの選択やベットの投入には，↑，↓，←，→，2，4，6，8，J，K，I，Mが使えます。また，決定には，リターンキーとスペースキーが使えます。各機種で使いやすいキーをご利用ください。ゲーム中，SHIFT＋BREAKでこのゲームを終わるかどうかを選択できます。"QUIT" で終了します。

そして，プレイヤーが勝ったとき以下の条件で配当がもらえます。

役	配当
・ブラックジャック（2枚ともスペード）	5倍
・ブラックジャック	3倍
・5カード	3倍
・6カード	4倍
・7カード	5倍
・スリーセブン（手札が7・7・7）	5倍
・以上のいずれでもないとき	2倍

プログラムについて

このプログラムは，SLANGで記述してあります。ソースにして約13Kバイトとけっこう大きくなってしまいました。GOTO文の使用や大域変数が多いなど，私がまだ

BASICのコーディングテクニックから抜けきれないところがあります。

よって，かなり読みづらいリストになってしまいましたが，コメントをそれなりに入れたので，プログラミング初心者には参考になるでしょう。

SLANGはタブコードに対応していないので，入力はタブコードに変換しないエディタを使用してください。1990年11月号のEDC-Tエディタしかない人は，3F68$_H$番地の値を00$_H$に変えることによって，タブコードに変換しなくなります。実行にはオフセットがつけてあります。コンパイル後，そのままでは動かないので，3000$_H$番地に転送してから実行してください。

また，コイン枚数とベット数は2バイト変数で管理されているので，，65535をこえると0に戻ります。まあ，そこまでいく人はいないでしょう。

リストの最初にあるCONST宣言で，各種定数を設定しており，この値を変えることにより各機種用に調整できます。普通の8ビットマシンの人は速すぎると思うので，定数WAITの値を増やしてみてください。

以下にいくつか変数を載せておきます。

WAIT………ゲームスピード
COINS……•起動時コイン枚数
MAX……….最大ベット数
BG…………背景キャラ

終わりに

今度は調子にのってポーカーやスピードなどのカードゲームから，はては麻雀（ちょっと無理かな？）などを制作するつもりです。期待していてください。

リスト1

```
  1: //    BLACK JACK for S-OS
  2: //        1992/11/25
  3: //  Copyright (C) K.Watanabe
  4: //
  5:
  6: org    $3000;
  7: offset $8000 - $3000;
  8:
  9: /*  ｺﾝﾌｨｸﾞﾚｰｼｮﾝ  */
 10: const WAIT  = 100,
 11:       COINS =  50,
 12:       MAX   =  10,
 13:       BG    =  '{';  /*  S-OSﾃﾞﾊ ﾍﾞﾀｷｬﾗ  */
 14:
 15: array byte card[] = ( "0A23456789TJQK" ),
 16:       byte fuda[51][1],
 17:       byte com[10][1],
 18:       byte you[10][1];
 19:
 20: var kidou, coin, ct, cd, bt,
 21:     cm, cs, yo, ys, wd,
 22:     bjc, bjy, com5, you5;
 23:
 24: main()
 25: (
 26:     var i, j, k;
 27:     wd = mem[$1f5c];
 28:     kidou = 0;
 29:     width(40);
 30: restart:
 31:     title();
 32:     print(str$(BG, 1000));
 33:     coin = COINS;
 34:     msg( 1,  0, "+--------------+");
 35:     msg( 1,  1, "l              l");
 36:     msg( 1,  2, "l  BLACK JACK  l");
 37:     msg( 1,  3, "l              l");
 38:     msg( 1,  4, "+--------------+");
 39:     msg(21,  0, "+----------------+");
 40:     msg(21,  1, "l   COINS   BETS l");
 41:     msg(21,  2, "l                l");
 42:     msg(21,  3, "l                l");
 43:     msg(21,  4, "+----------------+");
 44:     msg( 1,  6, "+------+");
 45:     msg( 1,  7, "lDEALERl");
 46:     msg( 1,  8, "+------+");
 47:     msg( 1, 16, "+------+");
 48:     msg( 1, 17, "lPLAYERl");
 49:     msg( 1, 18, "+------+");
 50:     /*  ｶｰﾄﾞ ｼｮｷﾁ  */
 51:     for i = 0 to 3 (
 52:         for j = 0 to 12 (
 53:             fuda[j + I * 13][0] = j + 1;
 54:             fuda[j + I * 13][1] = i + 1;
 55:         )
 56:     )
 57:     /*  ﾒｲﾝ ﾙｰﾁﾝ  */
 58:     ct = 0;
 59:     while(1) (
 60:         cd = 0; cm = 0; yo = 0;
 61:         shaful();
 62:         /*  ﾍﾞｯﾄ ｳｹﾂｹ  */
 63:         if ct == 0 then bt = bets(); else bt = ct;
 64:         hajime();
 65:         kazu();
 66:         /*  youﾉ Black Jackﾊﾝﾃｲ  */
 67:         bjy = 0;
 68:         if yousu() == 21 (
 69:             beep(); beep();
 70:             bjy = 1;
 71:             goto skp1;
 72:         )
 73:         /*  HIT ｼｮﾘ  */
 74:         while (stand() == 2) (
 75:             you[yo][0] = fuda[cd][0];
 76:             you[yo][1] = fuda[cd][1];
 77:             if yo == 4 (  /*  5ﾏｲｲｼﾞｮｳﾊ ｶｻﾈﾃ  */
 78:                 for i = 0 to 7 (
 79:                     locate(10, 16 + i);
 80:                     print(str$(BG, 30));
 81:                 )
 82:                 for i = 0 to 3
 83:                     drow(10 + i * 4, 16, you[i][0], you[i][1]);
 84:             )
 85:             if yo < 4
 86:                 drow(10 + yo * 7, 16, you[yo][0], you[yo][1]);
 87:             else
 88:                 drow(10 + yo * 4, 16, you[yo][0], you[yo][1]);
 89:             cd++; yo++; beep();
 90:             kazu();
 91:             if (ys = yousu()) >= 21 or yo == 7 exit;
 92:         )
 93: skp1:
 94:         /*  comﾉ ｼｺｳ  */
 95:         drow(17, 6, com[1][0], com[1][1]);
 96:         beep();
 97:         cs = comsu();
 98:         if ys > 21 or bjy == 1 goto skp2;
 99:         you5 = 0;
100:         if yo >= 5 and ys <= 21 then you5 = 1;
101:         while (ys > (cs = comsu()) or comsu() < 12) (
102:             timer();
103:             com[cm][0] = fuda[cd][0];
104:             com[cm][1] = fuda[cd][1];
105:             if cm == 4 (  /*  5ﾏｲｲｼﾞｮｳﾊ ｶｻﾈﾃ  */
106:                 for i = 0 to 7 (
107:                     locate(10, 6 + i);
108:                     print(str$(BG, 30));
109:                 )
110:                 for i = 0 to 3
111:                     drow(10 + i * 4, 6, com[i][0], com[i][1]);
112:             )
113:             cd++;
114:             beep();
115:             if cm < 4 (
116:                 drow(10 + cm * 7, 6, com[cm][0], com[cm][1]);
117:                 cm++;
118:             )else (
119:                 drow(10 + cm * 4, 6, com[cm][0], com[cm][1]);
120:                 cm++;
121:             )
122:         )
123: skp2:
124:         /*  comﾉ ｽｳ  */
125:         cs = comsu();
126:         msg(2, 10, "+----+");
127:         msg(2, 11, "l    l");
128:         msg(2, 12, "+----+");
129:         locate(4, 11);
130:         print(form$(cs, 2));
131:         /*  comﾉ Black Jackﾊﾝﾃｲ  */
132:         bjc = 0;
133:         if cs == 21 and cm == 2 then bjc = 1;
134:         com5 = 0;
135:         if cm >= 5 and cs <= 21 then com5 = 1;
136:         if yaku() == 1 goto restart;  /*  ﾔｸ ﾊﾝﾃｲﾍ  */
137:     )
138:     width(wd);
139: )
140:
141: // ｶｰﾄﾞﾉ ﾋｮｳｼﾞ
142: drow( x, y, c, m)
143: (
144:     var i;
145:     msg(x, y, "/----+");
146:     for i = 1 to 6 (
147:         msg(x, y + i, "l    l");
148:     )
149:     msg(x, y + 7, "+----/");
150:     case c of (
151:         0 : (
152:             msg(x + 1, y + 1, " xx ");
153:             msg(x + 1, y + 2, "x{xx");
154:             msg(x + 1, y + 3, "xx{x");
155:             msg(x + 1, y + 4, "x{xx");
156:             msg(x + 1, y + 5, "xx{x");
157:             msg(x + 1, y + 6, " xx ");
158:         )
159:         10 : (
160:             msg(x + 1, y + 1, "10");
161:             msg(x + 3, y + 6, "10");
162:         )
163:         others : (
164:         locate(x + 1, y + 1); print(str$(card[c], 1));
165:         locate(x + 4, y + 6); print(str$(card[c], 1));
166:         )
```

```
167:         )
168:         locate(x + 1, y + 3);
169:         case m of (
170:          0 : ;
171:          1 : print("ﾊｰﾄ");
172:          2 : print("ｸﾗﾌﾞ");
173:          3 : print("ﾀﾞｲﾔ");
174:          4 : (
175:         .        print("ｽﾍﾟｰ");
176:                  msg(x + 1, y + 4, "ﾄﾞ");
177:              )
178:         )
179: )
180:
181: //  ｶｰﾄﾞ ｼｬｯﾌﾙ
182: shaful()
183: (
184:     var a, b, i, dm0, dm1;
185:     for i = 1 to 100 + rnd(100) (
186:         a = rnd(52); b = rnd(52);
187:         dm0 = fuda[a][0];
188:         dm1 = fuda[a][1];
189:         fuda[a][0] = fuda[b][0];
190:         fuda[a][1] = fuda[b][1];
191:         fuda[b][0] = dm0;
192:         fuda[b][1] = dm1;
193:     )
194: )
195:
196: // ｶｽﾞ ﾋｮｳｼﾞ
197: kazu()
198: (
199:     ys = yousu();
200:     msg(2, 20, "+----+");
201:     msg(2, 21, "1.   1");
202:     msg(2, 22, "+----+");
203:     locate(4, 21);
204:     print(form$(ys, 2));
205: )
206:
207:
208: //  2ﾏｲｽﾞﾂ ｸﾊﾞﾙ
209: hajime()
210: (
211:         ct = 0;
212:     /*  youﾉ 1ﾏｲﾒ  */
213:     you[yo][0] = fuda[cd][0];
214:     you[yo++][1] = fuda[cd++][1];
215:     drow(10, 16, you[0][0], you[0][1]);
216:     beep(); timer();
217:     /*  comﾉ 1ﾏｲﾒ  */
218:     com[cm][0] = fuda[cd][0];
219:     com[cm++][1] = fuda[cd++][1];
220:     drow(10, 6, com[0][0], com[0][1]);
221:     beep(); timer();
222:     /*  youﾉ 2ﾏｲﾒ  */
223:     you[yo][0] = fuda[cd][0];
224:     you[yo++][1] = fuda[cd++][1];
225:     drow(17, 16, you[1][0], you[1][1]);
226:     beep(); timer();
227:     /*  comﾉ 2ﾏｲﾒ  */
228:     com[cm][0] = fuda[cd][0];
229:     com[cm++][1] = fuda[cd++][1];
230:     drow(17, 6, 0, 0);
231:     beep();
232: )
233:
234: //' ﾌﾟﾚｲﾔｰﾉ ｶｽﾞ ﾊﾝﾃｲ
235: yousu()
236: (
237:     var ace, c, i, sum;
238:     ace = 0; sum = 0;
239:     for i = 0 to yo - 1 (
240:         if (c = you[i][0]) == 1 then ace = 1;
241:         if c >= 10 then c = 10;
242:         sum = sum + c;
243:     )
244:     if ace == 1 and sum + 10 <= 21 return (sum + 10);
245:     return (sum);
246: )
247:
248: //  ｺﾝﾋﾟｭｰﾀﾉ ｶｽﾞ ﾊﾝﾃｲ
249: comsu()
250: (
251:     var ace, c, i, sum;
252:     ace = 0; sum = 0;
253:     for i = 0 to cm - 1 (
254:         if (c = com[i][0]) == 1 then ace = 1;
255:         if c>= 10 then c = 10;
256:         sum = sum + c;
257:     )
258:     if ace == 1 and sum + 10 <= 21 return (sum + 10);
259:     return (sum);
260: )
261:
262: //  Massage ﾋｮｳｼﾞ
263: msg(x, y, adr)
264: (
265:     locate(x, y);
266:     print(!(adr));
267: )
268:
269: //  ﾔｸ ﾊﾝﾃｲ
270: yaku()
271: (
272:     var i;
273:     msg(13, 14, "+----------+");
274:     msg(13, 15, "1          1");
275:     msg(13, 16, "+----------+");
276:     /*  ﾔｸ ﾊﾝﾃｲ  */
277:     if bjy == 1 (
278:         if you[0][1] == 4 and you[1][1] == 4 (
279:             msg(14, 15, "Black Jack");
280:             beep(); beep();
281:             ct = bt * 5;
282:         )elseif bjc == 0 (
283:             msg(14, 15, "Black Jack");
284:             ct = bt * 3;
285:         )else goto wake;
286:         continue();
287:         goto skp;
288:     )else if you5 == 1 and bjc == 0 (
289:         if com5 == 0 or (com5 == 1 and ys > cs ) (
290:             locate(15, 15);
291:             print(yo, " CARDS");
292:             ct = bt * (yo - 2);
293:         )else goto wake;
294:         continue();
295:         goto skp;
296:     )elseif you[0][0] == 7 and you[1][0] == 7
297:                 and you[2][0] == 7 and yo == 3 (
298:         msg(16, 15, "ｽﾘｰ 7");
299:         ct = bt * 5;
300:         continue();
301:         goto skp;
302:     )
303:     if ys == cs and bjc == bjy (   /*  ﾋｷﾜｹ ﾊﾝﾃｲ  */
304: wake:
305:         msg(15, 15, "= DRAW =");
306:         ct = bt;
307:         continue();
308:         goto skp;
309:     )
310:     if (ys <= 21 and ys > cs) or cs > 21 (
311:             msg(16, 15, "WIN !!");
312:             ct = bt * 2;
313:             continue();
314:     )else (
315:         if ys > 21 msg(16, 15, "BUST !");
316:         else msg(14, 15, "DEALER WIN");
317:         locate(32, 3);
318:         print(%(0));
319:         i = 0;
320:         if coin == 0 (
321:             msg(10,  8, "+---------------+");
322:             msg(10,  9, "1   GAME OVER   1");
323:             msg(10, 10, "1  ﾜﾊﾊ､ｵｹﾗﾀﾞ !  1");
324:             msg(10, 11, "+---------------+");
325:             repeat (
326:                 if i < 150 then i++;
327:             )until (inkey(0) != 0 and i >= 150);
328:             return (1);
329:         )
330:         key();
331:     )
332: skp:
333:     for i = 0 to 2 (
334:         locate(2, 10 + i);
335:         print(str$(BG, 6));
336:     )
337:     for i = 0 to 2 (
338:         locate(2, 20 + i);
339:         print(str$(BG, 6));
340:     )
341:     for i = 0 to 17 (
342:         locate(10, 6 + i);
343:         print(str$(BG, 30));
344:     )
345:     return (0);
346: )
347:
348:
349:
350: //  BETSﾄｳﾆｭｳ
351: bets()
352: (
353:     var k, bt, i;
354:     bt = 0;
355: bet_lp:
356:     msg(26, 10, "+-----------+");
357:     msg(26, 11, "1 BET PLEASE1");
358:     msg(26, 12, "1      <--> 1");
359:     msg(26, 13, "+-----------+");
360:     while(1) (
361:         locate(28, 12); print(form$(bt, 3));
362:         locate(32,  3); print(%(bt));
363:         locate(25,  3); print(%(coin));
364:         case inkey(2) of (
365:             '4', '¥l', 'J',
366:             'j', '2', '¥d',
367:             'M', 'm' : (
368:                 if bt > 0 (
369:                     bt--; coin++;
370:                 )
371:             )
372:             '6', '¥r', 'I',
373:             'i', '8', '¥u',
374:             'K', 'k' : (
375:                 if coin > 0 and bt < MAX (
376:                     bt++; coin--;
377:                 )
378:             )
379:             '¥n', ' ' : (
380:                 if bt > 0 exit;
381:             )
382:             $1b : (
383:                 quit();
384:                 gato bet_lp;
385:             )
386:         )
387:     )
388:     for i = 0 to 3 (
389:         locate(26, 10 + i);
390:         print(str$(BG, 13));
391:     )
392:     return (bt);
393: )
394:
```

```
395: //  STAND・HITｾﾚｸﾄ
396: stand()
397: {
398:     var i, k, s, ss;
399:     s = 1; ss = 2;
400: st_lp:
401:     msg(30, 10, "+-------+");
402:     msg(30, 11, "l  STANDl");
403:     msg(30, 12, "l  HIT  l");
404:     msg(30, 13, "+-------+");
405:     while (1) {
406:         msg(31, 10 + ss, " ");
407:         msg(31, 10 +  s, "*");
408:         k = key();
409:         if k == '¥n' or k == ' ' exit;
410:         if k == $1b {
411:             quit();
412:             goto st_lp;
413:         }
414:         ss = s;
415:         if s == 1 then s = 2; else s = 1;
416:     }
417:     for i = 0 to 3 {
418:         locate(30, 10 + i);
419:         print(str$(BG, 9));
420:     }
421:     return (s);
422: }
423:
424: //  CONTINUEｽﾙｶ?
425: continue()
426: {
427:     var i, k, s, ss;
428: ct_lp:
429:     msg(25, 10, " +-----------+");
430:     msg(25, 11, " l CONTINUE  l");
431:     msg(25, 12, " l TAKE SCOREl");
432:     msg(25, 13, " +-----------+");
433:     locate(32, 3);
434:     print(%(ct));
435:     s = 1; ss = 2;
436:     while (1) {
437:         msg(27, 10 + ss, " ");
438:         msg(27, 10 +  s, "*");
439:         k = key();
440:         if k == '¥n' or k == ' ' exit;
441:         if k == $1b {
442:             quit();
443:             goto ct_lp;
444:         }
445:         ss = s;
446:         if s == 1 then s = 2; else s = 1;
447:     }
448:     if s == 2 {
449:         for i = ct - 1 downto 0 {
450:             locate(25, 3);
451:             print(%(++coin));
452:             if inkey(0) != 0 and ct - i > 1 {
453:                 coin = coin + i;
454:                 i = 0;
455:             }
456:             locate(32, 3);
457:             print(%(i));
458:             beep();
459:         }
460:         ct = 0;
461:         locate(25, 3);
462:         print(%(coin));
463:     }
464: }
465:
466: //  INKEYｶﾝｽｳ
467: key()
468: var kk = 0;
469: {
470:     var k;
471:     repeat {
472:         rnd(100);
473:         k = inkey(0);
474:         if k == 0 then kk = 0;
475:     } until (k != 0 and k != kk);
476:     kk = k;
477:     return (k);
478: }
479:
480: //  ｼﾞｶﾝﾏﾁ ﾙｰﾌﾟ
481: timer()
482: {
483:     var i, t;
484:     for t = 0 to WAIT {
485:         for i = 1 to 100
486:             ;
487:     }
488: }
489:
490: //  ｷｮｳｾｲ ｼｭｳﾘｮｳ
491: quit()
492: {
493:     var i, k, s, ss;
494:     s = 1; ss = 2;
495:     msg(27, 12, "+--------+");
496:     msg(27, 13, "l  CANSELl");
497:     msg(27, 14, "l  QUIT  l");
498:     msg(27, 15, "+--------+");
499:     while (1) {
500:         msg(28, 12 + ss, " ");
501:         msg(28, 12 +  s, "*");
502:         k = inkey(2);
503:         if k == '¥n' or k == ' ' exit;
504:         if k == $1b {
505:             s = 1;
506:             exit;
507:         }
508:         ss = s;
509:         if s == 1 then s = 2; else s = 1;
510:     }
511:     if s == 2 {
512:         width(wd);
513:         stop();
514:     }
515:     while (inkey(0) != 0) ;
516:     for i = 0 to 3 {
517:         locate(27, 12 + i);
518:         print(str$(BG, 10));
519:     }
520: }
521:
522: //  ﾀｲﾄﾙ ﾋｮｳｼﾞ
523: title()
524: {
525:     var k;
526:     if kidou++ == 0 print(str$(BG, 1000));
527:     locate(0, 24);
528:     print(cr$(4));
529:     print("   /---+  l         ^       ---  l  /", /);
530:     print("   l   l  l       l l   l       l / ", /);
531:     print("   l--<   l      l---l  l       l/   ", /);
532:     print("   l   l  l      l   l  l       l l ", /);
533:     print("   +---/  +---   l   l   ---   l  L", /);
534:     print(/, /);
535:     msg(6, 24, "    T    ^      ---  l  /    l  l¥n");
536:     msg(6, 24, "    l   l l   l      l /     l  l¥n");
537:     msg(6, 24, "    l  l---l  l      l/      l  l¥n");
538:     msg(6, 24, "l   l  l   l  l      l l        ¥n");
539:     msg(6, 24, "+---/  l   l   ---   l  L    *  *¥n");
540:
541:     print(cr$(9));
542:     msg(9, 17, "Copyright (C)  K chan");
543:     msg(5, 21, " = PUSH SPACE KEY TO START =");
544:     while (1) {
545:         k = key();
546:         if k == 'Q' or k == 'q' or k == $1b {
547:             width(wd);
548:             stop();
549:         }
550:         if k == ' ' or k == '¥n' exit;
551:     }
552:     locate(0, 24);
553: }
554:
555: /* モニターありがとう！
556:       市川君  大窪君
557:       笹島君  森君
558:       高須君  吉田君
559: */
```

▶ 全 機 種 共 通 シ ス テ ム イ ン デ ッ ク ス ◀

＊以下のアプリケーションは，基本システムであるS-OS“MACE”またはS-OS“SWORD”がないと動作しませんのでご注意ください。

1986

■86年1月号
第13部 リロケータブルのお話
第14部 FM音源サウンドエディタ
■86年2月号
第15部 S-OS "SWORD"
第16部 Prolog-85入門(1)
■86年3月号
第17部 magiFORTH発表
連載 Prolog-85入門(2)
■86年4月号
第18部 思考ゲームJEWEL
第19部 LIFE GAME
連載 基礎からのmagiFORTH
連載 Prolog-85入門(3)
■86年5月号
第20部 スクリーンエディタE-MATE
連載 実戦演習magiFORTH
■86年6月号
第21部 Z80TRACER
第22部 magiFORTH TRACER
第23部 ディスクダンプ&エディタ
第24部 "SWORD" 2000 QD
連載 対話で学ぶmagiFORTH
特別付録 PC-8801版S-OS "SWORD"
■86年7月号
第25部 FM音源ミュージックシステム
付録 FM音源ボードの製作
連載 計算力アップのmagiFORTH
特別付録 SMC-777版S-OS "SWORD"
■86年8月号
第26部 対局五目並べ
第27部 MZ-2500版S-OS "SWORD"
■86年9月号
第28部 FuzzyBASIC発表
連載 明日に向かってmagiFORTH
■86年10月号
第29部 ちょっと便利な拡張プログラム
第30部 ディスクモニタDREAM
第31部 FuzzyBASIC料理法<1>
■86年11月号
第32部 パズルゲームHOTTAN
第33部 MAZE in MAZE
連載 FuzzyBASIC料理法<2>
■86年12月号
第34部 CASL & COMET
連載 FuzzyBASIC料理法<3>

1987

■87年1月号
第35部 マシン語入力ツールMACINTO-C
連載 FuzzyBASIC料理法<4>
■87年2月号
第36部 アドベンチャーゲームMARMALADE
第37部 テキアベ作成ツールCONTEX
■87年3月号
第38部 魔法使いはアニメがお好き
第39部 アニメーションツールMAGE
付録 "SWORD" 再掲載とMAGICの標準化
■87年4月号
第40部 INVADER GAME
第41部 TANGERINE
■87年5月号
第42部 S-OS "SWORD" 変身セット
第43部 MZ-700用 "SWORD" をQD対応に
■87年6月号
インタラプト コンパイラ物語
第44部 FuzzyBASICコンパイラ
第45部 エディタアセンブラZEDA-3
■87年7月号
第46部 STORY MASTER
■87年8月号
第47部 パズルゲーム碁石拾い
第48部 漢字出力パッケージJACKWRITE
特別付録 FM-7/77版S-OS "SWORD"
■87年9月号
第49部 リロケータブル逆アセンブラInside-R
特別付録 PC-8001/8801版S-OS "SWORD"
■87年10月号
第50部 tiny CORE WARS
第51部 FuzzyBASICコンパイラの拡張
第52部 X1turbo版S-OS "SWORD"
■87年11月号
序論 神話のなかのマイクロコンピュータ
付録 S-OSの仲間たち
第53部 もうひとつのFuzzyBASIC入門
第54部 ファイルアロケータ&ローダ
インタラプト S-OSこちら集中治療室
第55部 BACK GAMMON
■87年12月号
第56部 タートルグラフィックパッケージTURTLE
第57部 X1turbo版 "SWORD" アフターケア
ラインプリントルーチン
特別付録 PASOPIA7版S-OS "SWORD"

1988

■88年1月号
第58部 FuzzyBASICコンパイラ・奥村版
付録 石上版コンパイラ拡張部の修正
■88年2月号
第59部 シューティングゲームELFES
■88年3月号
第60部 構造型コンパイラ言語SLANG
■88年4月号
第61部 デバッギングツールTRADE
第62部 シミュレーションウォーゲームWALRUS
■88年5月号
第63部 シューティングゲームELFES II
第64部 地底最大の作戦
■88年6月号
第65部 構造化言語SLANG入門(1)
第66部 Lisp-85用NAMPAシミュレーション
■88年7月号
第67部 マルチウィンドウドライバMW-1
連載 構造化言語SLANG入門(2)
■88年8月号
第68部 マルチウィンドウエディタWINER
■88年9月号
第69部 超小型エディタTED-750
第70部 アフターケアWINERの拡張
■88年10月号
第71部 SLANG用ファイル入出力ライブラリ
第72部 シューティングゲームMANKAI
■88年11月号
第73部 シューティングゲームELFES IV
■88年12月号
第74部 ソースジェネレータSOURCERY

1989

■89年1月号
第75部 パズルゲームLAST ONE
第76部 ブロックゲームFLICK
■89年2月号
第77部 高速エディタアセンブラREDA
特別付録 X1版S-OS "SWORD"<再掲載>
■89年3月号
第78部 Z80用浮動小数点演算パッケージSORBAN
■89年4月号
第79部 SLANG用実数演算ライブラリ
■89年5月号
第80部 ソースジェネレータRING
■89年6月号
第81部 超小型コンパイラTTC
■89年7月号
第82部 TTC用パズルゲームTICBAN
■89年8月号
第83部 CP/M用ファイルコンバータ
■89年9月号
第84部 生物進化シミュレーションBUGS
■89年10月号
第85部 小型インタプリタ言語TTI
■89年11月号
第86部 TTI用パズルゲームPUSH BON!
■89年12月号
第87部 SLANG用リダイレクションライブラリDIO.LIB

1990

■90年1月号
第88部 SLANG用ゲームWORM KUN
特別付録 再掲載SLANGコンパイラ
■90年2月号
第89部 超小型コンパイラTTC++
■90年3月号
第90部 超多機能アセンブラOHM-Z80
■90年4月号
第91部 ファジィコンピュータシミュレーションI-MY
■90年5月号
第92部 インタプリタ言語STACK
■90年6月号
第93部 リロケータブルフォーマットの取り決め
第94部 STACK用ゲームSQUASH!
第95部 X68000対応S-OS "SWORD"
特別付録 PC-286対応S-OS "SWORD"
■90年7月号
第96部 リロケータブルアセンブラWZD
■90年8月号
第97部 リンカWLK
■90年9月号
第98部 BILLIARDS
■90年10月号
第99部 ライブラリアンWLB
■90年11月号
第100部 タブコード対応エディタEDC-T
■90年12月号
第101部 STACKコンパイラ

1991

■91年1月号
第102部 ブロックアクションゲームCOLUMNS
■91年2月号
第103部 ダイスゲームKISMET
■91年3月号
第104部 アクションゲームMUD BALLIN'
■91年4月号
第105部 SLANG用カードゲームDOBON
■91年5月号
第106部 実数型コンパイラ言語REAL
■91年6月号
第107部 Small-C処理系の移植
■91年7月号
第108部 REALソースリスト編
■91年8月号
第109部 Small-Cライブラリの移植
■91年9月号
第110部 SLANG用NEWファイル出力ライブラリ
■91年10月号
第111部 Small-C活用講座(初級編)
■91年11月号
第112部 Small-C活用講座(応用編)
第113部 MORTAL
■91年12月号
第114部 Small-C SLANGコンパチ関数

1992

■92年1月号
第115部 LINER
■92年2月号
第116部 シミュレーションゲームPOLANYI
■92年3月号
第117部 カードゲームKLONDIKE
■92年4月号
第118部 オプティマイザO80実践Small-C講座(1)
■92年5月号
第119部 COMMAND.OBJ実践Small-C講座(2)
■92年6月号
第120部 COMMAND.OBJ2実践Small-C講座(3)
■92年7月号
第121部 関数リファレンス実践Small-C講座(4)
■92年8月号
第122部 ワイルドカード実践Small-C講座(5)
第123部 グラフィックライブラリ GRAPH.LIB
■92年9月号
第124部 O-EDIT&MODCNV
■92年10月号
第125部 SLENDER HUL実践Small-C講座(6)
■92年11月号
第126部 EDIT実践Small-C講座(7)
■92年12月号
第127部 MAKE実践Small-C講座(8)

CREATIVE COMPUTER MUSIC

Creative Computer Music入門(17)
金管楽器の基礎知識

先月紹介した金管楽器ですが，そのなかで最もよく使われるトランペットとトロンボーン，チューバの特徴などを，今月はもう少し詳しく説明します。DTMの技巧などには直接の関係はありませんが，楽器の特性を知ることは，よりリアルな作曲やアレンジに応用できるでしょう。

Taki Yasushi 瀧 康史

§聴き比べてほしいPOWER

まず最初にCDを紹介しましょう。今月選んだCDはTOWER OF THE POWER。金管楽器について考えるうえで，いいお手本になると思います。

さて，TOWER OF THE POWERでピンとこなくても，ヒューイ・ルイスのバックバンドを務めていたといえば，わかる人もいることでしょう。サウンドは，ボーカルのあるブラス主体のバンドなのですが，先月のスペクトラムが和風のさっぱり味なら，こっちはちょっとダークさがあるいい感じってところ。スペクトラムをロックがベースのバンドというなら(ブラスロックの伝説のバンドといわれています)，こちらは対照的にファンクがベースといえるでしょう。その点について聴き比べると面白いと思います。バンド自体の活動歴は長くて，結成から20年ぐらい，レコードデビューから実に17年もたっています。積もり積もった技術は大きいといいますが，まさにそのとおりです。ブラスの音の厚みは，アルト，テナー，バリトンの3本のサックスのパワーも加わって(サックスは木管楽器です。念のため)，「音の壁が押し寄せてくる」という形容がピッタリなほど。

機会があれば，CD「TOWER OF THE POWER」に入っている「BOYS NIGHT OUT」と，1974年に発売された「BACK TO OAKLAND」の中の「SQUIB CAKES」と，1991年のアルバム「MONSTER ON A LEASH」の「Mr. TOAD'S WILD RIDE」の3曲のインストゥルメンタルを聴き比べてほしいですね。

ブラスサウンドはどちらかというと乾いたイメージがあって，寒い季節には似合わない感じもしますが，力の塔(TOWER OF THE POWER)という名前はダテではありません。おそらく，18禁の世界を味わえることでしょう(嘘800！)。

§金管楽器のバランス

前回に引き続いて，今回も金管楽器について話を進めていきます。ただし今回は，前回よりも，蘊蓄(うんちく)に近いかなり突っ込んだ話になりますから，そのつもりで。

どちらかというと今回は「生のブラス」を使うようなイメージで話を進めていきますが，これらの知識はDTMでもそれなりに役に立つと思います。シンセブラスについては，これとは別にシンセサイザを説明するときに一緒に行います。

さて今回は，金管楽器のなかでも特にトランペット，トロンボーン，チューバの3つを扱うことにします。ホルンは，確かに金管楽器ですが，木管楽器と同時にアンサンブルを考えたほうがわかりやすいので，今回は除外します。ホルンの音のファンの人には，しばらくお預けということでごめんなさい。木管楽器について解説するときまで，がまんしてくださいね。

ではまず最初に，完全編成ともいえる管弦楽での，金管楽器の内訳をみてみましょう。以下のものが一般的な内訳です。

ホルン	×	4
トランペット	×	3
トロンボーン	×	3
		(テナー×2，バス×1に分類)
チューバ	×	1

もちろん，これらの数は一般的な基準であって，それ以上の楽器を使った管弦楽曲もたくさんあります。ワーグナー，シュトラウス，ストラヴィンスキー，ホルストなどの曲を聴けば，もっとたくさんの金管楽器を使った曲に出会えるでしょう。

この数字がどこから出てきたかというと，スキのないアンサンブルを作るための最小限編成と，演奏者を集めるうえでの経済的な理由の兼ね合いからです。しかし，DTMでは演奏者を雇う資金などは考えなくてもよいので(とはいってもそれなりのMIDI楽器を買う資金は必要ですが)，DTMの場合の限界は，楽器の同時発声数の限界ということになるでしょう。

金管楽器は重音はできませんので，これらの数は同じ音ならば，ある程度の本数まではMIDI 1chで重音させてすますことができます。したがって，ポリフォニックのMIDI楽器を使ったときの構成は次のとおりといえるでしょう。

ホルン	×	2 ch
トランペット	×	2 ch
トロンボーン	×	2 ch
チューバ	×	1 ch

しかし，これに限定されず，できる限り音数を削らずに複数の声部を同時発声させることを考えなくてはいけません。MIDIチャンネルは1つにつき1つのボリュームしか設定できないでしょうし，MMLを使った音源ドライバの場合，同じ音での2つのメロディーパートを1chにまとめることは至難の業ですから，個人個人でそれらの長所短所をよく踏まえ，手持ちの楽器に合ったチャンネル設定をする必要があります。

また，各々の楽器は音色や音量などの点でそれぞれ特徴があります。したがって生演奏の場合，これらの音を同時に鳴らすとき，ちょうどキレイにハモらせるためにはある程度の配慮が必要になります。

たとえば，トランペットやトロンボーンが1本で，メゾフォルテ(「やや強く」)以上の大きな音で演奏して，同時にホルンを鳴らすとしましょう。その場合，ホルンのその丸みのある音色や音量の特徴のために，ホルン1本だけでは負けてしまって表面に音が出てこないのです。

オーケストラを組むとき，メゾフォルテ(mf)より弱い部分ならば

トランペット	：	トロンボーン	：	ホルン
1	：	1	：	1

で十分なアンサンブルのバランスがとれますが，メゾフォルテ以上の音量になると，

トランペット：トロンボーン：ホルン
1　：　1　：　2
という比率になります。そして，このホルンはユニゾンで演奏するのです。

したがって，もし最初に記した編成ならば，メゾフォルテ以上になるとホルンは2本しかないので，2重のインターバルしか演奏できなくなってしまいます。

もちろん，このようなことはDTMでは直接は関係ないかもしれませんが，プリセットサンプラーのようなMIDI楽器，たとえばありふれたCM-64やSC-55などを使う場合，ここにサンプリングされているホルンがどのようなバランスで録音されているか，注意をしなくてはいけません。

§トランペットの使い方

現在最もよく使われているトランペット(図1)はB♭の楽器で，そのため，ピアノ譜などとは違い，譜面上の記音は実際の音と異なります。下一線(通常O4Cのある部分)が，O3B♭であることは，以前お話ししたとおりです。

DTMでは，わざわざB♭の楽譜を書く必要はないでしょうが，トランペットの楽譜を読むときには必要な知識ですので，覚えておいてください。MMLの場合，譜面上の音をそのまま打ち込んで頭に長2度(全音)低く演奏する命令を加えると同じことができます。人によっては楽かもしれません。Z-MUSICでは「-k2」でOKです。

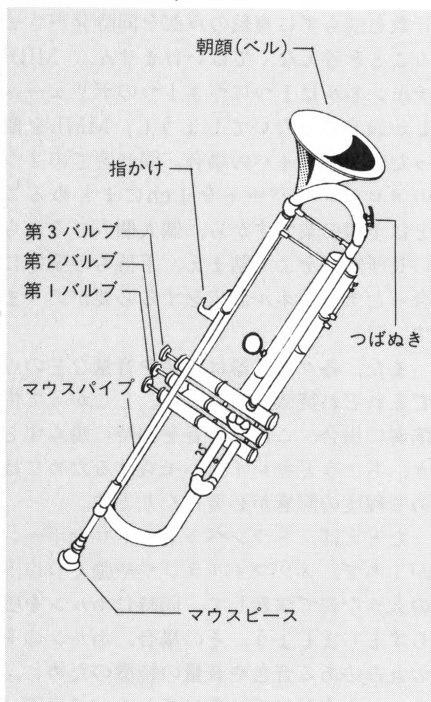



図1　トランペット

図2　トランペット〔B♭〕の音域

B♭管のトランペット（以下，単にトランペットといいます)の音域は図2のとおりですが，このなかでもO5E♭～O5A♭音は特別に扱うのが賢明です。

この部分のトランペットの音は非常に気高く，高貴で，鋭さがあります。したがってここいちばんのハイライトに使うべきでしょう。全体的にこの部分を使わずに控えめに曲を作り(もしくはアレンジし)，ここいちばんの盛り上がりでこの気高く鋭い音が使われると，非常に効果的です。

また，これより下の音は，演奏者が楽に演奏できるので，ppp(ピアニッシシモ)からfff(フォルティッシシモ)まで容易に，しかも確実に扱える音になります。

トランペットは軽快な楽器ですが，速くても6度以上に長い経過句(グリッサンド)は書かないほうが賢明です。これはバルブの特質に由来しています。5度の経過句は今回紹介したCD「TOWER OF THE POWER」の7曲目に頻繁に登場してきますので，注意して聴いてみるといいでしょう。

また，管楽器の特徴的な演奏方法にタンギング(シングル，ダブル，トリプルがある)があります。これによって，速くて明確なスタッカートを演奏することができます。

これは，同音で速い反復連符(16分音符の連続や6連符など)をするとき，きわめて強い印象を効果的に与えます。先月紹介したスペクトラムでも，この技法はところどころで使用されていますし，今月紹介したPOWERもまた同様に使用しています。

DTMでは直接関係がありませんが，もし，生で誰かに吹いてもらうつもりならば，速いタンギングを必要とする箇所ではどこかで息つぎのタイミングを与えなければなりません。そうでないと奏者は死んでしまいます……。逆にいえば，DTMでもわざとそういうタイミングをうまく意識的に置くことにより，それらしさを表現できるかもしれません。これらは各自の検討事項にしてください。

トランペットにミュート(カップ上の媒体を円錐形の口金に付けたもの)を付けて柔らかく吹くと，甘く，遠くから聴こえるような叙情的な音が出ます。シンセサイザによっては，これが「mute」という名前で入っています。この音は，強く吹くと鼻にかかったような音になりますが，弱く吹くとかなり穏やかに聴こえます。

近年，ミュートを付けて大きな音を出す，ということをいろいろな人がいろいろな曲で使ったため，いまではいまいちインパクトに欠けますが，演奏会などに行く機会があったら注意して聴いてみるといいかもしれません。

§トロンボーンの特徴

トロンボーンもB♭管の楽器ですが，バルブがなく，管をスライドさせることによってその長さを調節し，音を変化させる楽器です(図3)。このあたりの詳しいことは，先月号や今月号のコラムを読んでもらうことにして，ここでは主にトロンボーンの結合について考えてみましょう。

一般によく使われるトロンボーンには2種類あり，B♭管のものをテナートロンボーン，G管のものをバストロンボーンといいます。それぞれの音域を示したのが図4です。

トロンボーンはその「スライドにより音の高さを変化させる」という性格上，素早い演奏をするには限度があります。したがって，トップメロディを奏でさせることはあまりできないので，必然的に3声のアンサンブルで和声に撤し，その音の高さが中間的な位置にあるということから，和声的な結合役をまかせることになります。

トロンボーンはテナー2つにバス1つの3つを使い，この3つで和音を奏でるのですが，美しい音色，微妙な強弱，それらの理由から，これらの醸し出すハーモニーの効果はきわめて美しく，かつ印象的です。

通常，金管楽器では密集配置をするのが鉄則ですが，柔らかめの和音(強弱はmpからppが効果的)にしたいなら，トロンボーンの3和音は開離配置にし，低めの配置にするのが効果的です。

逆に曲のハイライトで，全体的にテンション(和声のテンションを指すのではない)を上げたいなら，高めに，mf以上のヴォリュームで密集か開離で置くのが効果的でしょう。

楽器の特質上，非常に肺活量が必要なので，どうしても長め(そうでなくても)fp(フォルテピアノ：「フォルテのあとにすぐにピアノに」という意味)になってしまうか，さらに長いなら，f>になってしまいます。

そのかわり，断続的なf，そして特にffのついた和音はリズミカルな，言い方を変え

るとかなり打撃的な効果をもち，これはかなり利用できます。

ドヴォルザークの「新世界から」や，最近ではジョン・ウイリアムズの「スターウォーズ」などのフィナーレがあまりにも強烈で有名なため,「トロンボーンは強烈な盛り上がりをもたせる楽器」というイメージが先行しがちですが，実は，この楽器のppでの演奏能力は，非常に高貴で和声的に豊かなものです。この柔らかな美しい和音は，木管楽器，弦楽器，ホルンなどのソロのバックに美しい背景を用意することができるのです。

トロンボーンにミュートを付けて強く吹くと，トランペット同様，鼻に詰まったような音になります。ユーモラスで奇怪な音なのですが，これもまたトランペットと同様，すでにダンスミュージックなどであまりにも使われすぎてしまっていて，もはや使い古しのギャグみたいに，効果は薄いといえるかもしれません。

かつては，トロンボーンのその性質から特徴的なグリッサンドが，いろいろな意味で妙なアクセントになったのですが，もはやこれもみんなが使いすぎて，インパクトが薄くなってしまったのも事実といえるでしょう。

しかし，ミュートを付けたときの，pもしくはppの長めの和音は，「酔いしれる」という言葉が似合うほど美しいハーモニーをもっています。私はこの上に，ホルンのソロ，フルートの穏やかなソロ，または逆にトリッキーなソロをのせるのが好きなのですが，(そうやって遊ぶことができる環境にいるなら)ぜひやってみることをオススメします。

§チューバの音色

その大きななりから誤解されがちですが，チューバというのは多彩で，うまく使えばたいへん利用価値がある楽器です。バルブを使う楽器のなかでは，唯一，４つのバルブをもつので音域は広く，またその音色は低音から高音まで非常に多彩です。

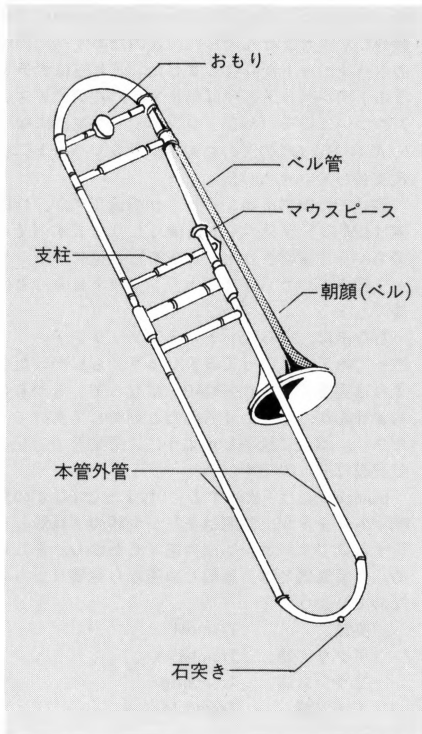
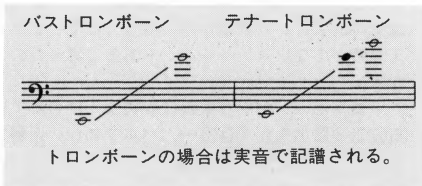
しかし，チューバの高音は非常に硬く，金属的で，激しい気性の音です。そのため，特殊なソロ以外では使わないほうが無難ともいえるでしょう。しかし，あまり使い古されていないので，うまく使えばなかなか効果的かもしれません。

チューバの目立ちすぎる音は利用価値が多く，低音部の経過的な音の連結を引き立たせるにはもってこいなのですが，これらは多用するとその価値が半減するので，いくらか控えめに利用するべきです。おいしいものは取っておくという気持ちで。

チューバの音は，柔らかくても鋭くても，トランペットやトロンボーン群との結合に強く，確実な低音を与えます。ただし，これらは正確な使われ方をしたときの話で，もしもその使い方を誤れば，チューバの音は著しく浮いてしまいます。

たとえば，mp以上の場合は，その音色の特質から，トランペット・トロンボーン群と音色的にうまく結合しないため，バス進行をバストロンボーンにとらせ，１オクターブ下でユニゾンをするなど，ある程度の考慮が必要です。

チューバはpもしくはpp程度で柔らかく吹くとホルンとよく結合します。しかし，これらの音は本質的にはまったく違う系統の音ですから，強く吹けば吹くほど，音質が分離しがちなので，ホルンとの融合はpまでと強く意識したほうがいいでしょう。

もちろん，多くのほかの楽器がここに入るときは，チューバとホルンの音質の隔たりをトロンボーンなどがうまく埋めてくれるため，これは有効です。こういうときは，チューバの音を若干低めにし，バス進行をすることで,よりハーモニーが結合できます。

チューバに限らず，全体的に金管楽器のハーモニーの鉄則はその和声が「柔らか」でありたいか，「鋭く」ありたいかで大きく違ってきます。

やわらかく結合されるときは，トランペットは鋭くなりがちな高音を避け，トロンボーンは開離配置で，その下でチューバはその独特的な柔らかく優しい音を独立に用いてバス進行を行えば，ハーモニーは豊かに融け合います。

逆に鋭く結合させるとき(派手にしたい場合でも和声的には結合しなくてはいけません)には，トランペットはその特徴ともいうべき甲高い高音の，シャリシャリしたアクセントの強い音を前面に押し出し，トロンボーンは若干高めの密集配置主体で結合させます。そして，チューバはこのままではうまく結合できないため，バストロンボーンとオクターブ下でユニゾンします。

チューバはあの大きさながら，軽めのスタッカートも行えますが，音質がほかの金管楽器とはかけ離れるため，うまく融合す

図3 トロンボーン

図4 トロンボーンの音域

図5 チューバ

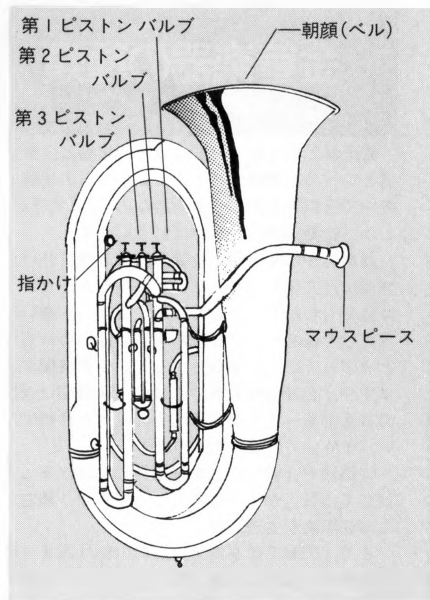
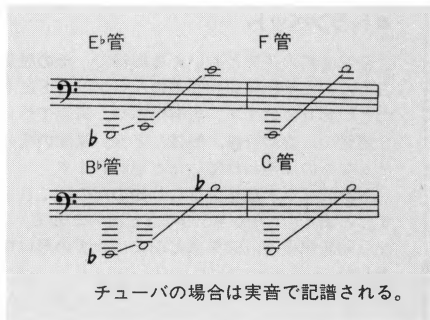



図6 チューバの音域

るには演奏者自身の熟練が必要になります。DTMの場合はこれらの心配はないでしょうが，やはり，結合の面で微妙なバランスが必要になりますから，慣れないうちはチューバの音を低めにして無難にとるほうがよいでしょう。

§曲のなかから

シンセサイザにプリセットされているブラスの音は，どちらかといえばアタックが長めです。甘いトランペット・ソロなどから，イメージがこびりついているために，アンサンブルに不向きな音色が多いのも事実です。

今月紹介したPOWERにおいても，先月紹介したスペクトラムでも，最近の日本でブラス系のポップスバンドとして有名な「米米クラブ」にしても，よく聴いてみたとき，ブラスのアタックは弱いでしょうか。

たいていは，歯切れのよさがアクセントをとって効果的に用いられていますよね。むしろ，一般的にはスラーがかっているよりも，タンギングがかっているブラスのほうが(特にポップスなどでは)よく使われています。

ブラスのパートの入るところも，曲を聴きながら読み取らねばならないところでしょう。たとえば，歌があるところには，どのようなフレーズでブラスが入ってくるのか，どのようなタイミングで使われるのか，ほかの楽器とどのように融け合うのか，そのような点をよく注意して聴いてみるべきでしょう。

ブラスは曲のハイライトで登場してくるパターンが多いので印象が強いのですが，それ以外のところでは，実際はどのような箇所でどのように使われているのか，案外みんなきちんと認識していないようです。たとえば，1992年8月号で紹介した「シンフォニーソーサリアン」では，ハイライト以外に，弦楽器や木管楽器が長く伸ばす音のときにまるで合いの手のように出てきます。これらはブラス楽器の典型的な用い方ですから，注意して聴いてみましょう。

では，これらをDTMで再現するにはどうすればよいでしょうか？　たとえばうまくゲートタイムを設定してみる。もっとも，MMLではその性格上，一音一音，音の長さを変えるのは大変かもしれません。しかし，平坦に8分音符が続くからといって，そのまま8分で演奏してしまうのではあまりに興ざめです。楽譜に隠れた情報をうまく引き出して再現する必要があるでしょう。それができないのなら，それはブラスではなく，ブラスの音のシンセサイザでしかないわけですから。

ここで文章で伝えてもなかなかピンとこないと思います。各自が実際に曲を聴いてみて，それをできる限り忠実に模倣してみるなど，それらを検討事項としてください。

§CM-64とSC-55

ではここで，わりとポピュラーな楽器，CM-64とSC-55の2つに入っているブラスの音色について，ちょっとだけ触れておきましょう。

最近はSC-55に株を奪われてしまった感

金管楽器の種類

管楽器というと，一般には木管楽器と，金管楽器というのを想像しますよね。どちらの楽器も管の中で空気を震動させる楽器なので，「気鳴楽器」とか「吹奏楽器」とも呼ばれています。

管楽器を木管楽器と金管楽器の2種に分ける分類法は，文字どおり，その管の素材からというのは誰でもわかることでしょう。しかし，違いはそれだけではありません。木管楽器はいわゆる「笛」と呼ばれていて，楽器そのものに発音機関がありますが，金管楽器の場合には，発音機関は演奏者の口唇であって，楽器そのものには発音機関がない，そういった違いもあります。

管楽器を金管楽器と木管楽器に分類することに対しては賛否が分かれるところですが，現在ではこの分類法が主流です。

さて，DTMでは楽器そのものの話はあまり関係ないかもしれませんが，知っていてソンな話ではないので，ちょっと説明しておきましょう。今回は本文との関連で金管楽器についてです。

●トランペット

金管楽器の花形ともいえる楽器で，その歴史は古く，およそ紀元前2000年頃からエジプトにあったといわれています。当時は木や，青銅で作られた直管で，宗教行事，競技，宴会，軍隊の信号，そんなものに使われていたと思われます。

トランペットが巻型になり始めたのはこれよりずっとあとの，今から300年ほど前のことで，これから開発が進み，150年ほど前にはいまの形になりました。

口唇とピストンの操作で音程を変える楽器ですが，むかしはこのピストンがなかったために，倍音列しか出せませんでした。このためすべての調のトランペットが存在しました。それらはナチュラルトランペットと呼ばれますが，次第にピストンのついているバルブトランペットが主流になり，F, A, B♭およびCの管のバルブ型トランペットに世代交替していきました。

現在ではB♭のトランペットが普通ですが，作曲家はC管のトランペットを使ったりします（それでもB♭管で演奏されたりします）。身近にトランペット吹きがいたら，そのトランペットはきっとB♭管でしょう。

ちなみに，彼らに「ドレミファソラシド～って歌ってみて」といってみましょう。もしかしたらそれは実音で「B♭CDE♭FGAB♭」になっているかもしれませんが，そういうものだと納得してあげてください。本文で説明したように，管楽器では記音と実音は違うのです。

「trumpet」には「吹聴する」「じょうご」などの意味がありますが，語源はギリシヤ語の「貝殻」を表す言葉です。ホルンは角笛でしたから，さしずめ，金管楽器とは，貝殻と角笛から発展したものなのでしょうか？

英語	Trumpet
イタリア語	Tromba
フランス語	Tronpette
ドイツ語	Trompete

●トロンボーン

きっと先月のガイーヌの「剣の舞」の「ぱぁ～あぱぁ～あ」という表現が頭に残ってるでしょう？え，残ってない？　インパクトあると思ったのになぁ。

トロンボーンという楽器は現在よく使われている金管楽器のなかでは唯一，バルブのない楽器です。音程の高さはフレックスに動作可能なスライドによって変わります。それで，グリッサンドができ，「ぱぁ～あ」ということができるのです。

トロンボーンは管弦楽ではトランペットの下の(低い)音程を支えるための楽器で，いまから600年ほど前にトランペットから生まれました。いままでは，ジャズなんかでもよく使われているため，耳にする機会が多いと思います。

語源はトランペットと同じで，トランペットが「小さなラッパ」という意味に対して，トロンボーンは「大きなラッパ」という意味です。

英語	Trombone
イタリア語	Trombone
フランス語	Trombone
ドイツ語	Pasaune

●ホルン

同じ金管楽器でも，ホルンはその音質から，木管楽器と同様にアンサンブルを組むほうが楽だという人もいます(私もこの考え方です)。実際，音色も似ているため，金管楽器でありながら木管演奏に加えられることもしばしばです。

一般によくいわれるホルンは，フレンチホルンのことで，これはどう見ても金管楽器です。もともとホルン(horn)は「角笛」のことを表していて，ヨーロッパでは，羊飼いや，郵便配達夫の持っていたラッパもホルンといわれています。アルペンホルンが角笛の系統だということは，誰でもすぐに気がつくでしょうけど。ちなみに，ホルン(horn)とはドイツ語で，これに対応する英語ではフォーン(phone)。ただし，楽器のホルンは英語でもhornです。

それから，イングリッシュホルン。

友達にいわれるまで気がつかなかったのですが(おおまぬけ)，これはまっとうな木管楽器です。私は曲を作るときにも使ったことがなかったので

じのするCM-64ですが，中には結構な数のブラスが入っています。この楽器のPCM部分は，ほとんどピアノ，ベース，オルガン，ブラスが重要視されてしまっているので，これが第一線を離れてしまった原因といえるでしょう。曲のデータを作るにも，たいていは何かカードが必要になってしまいますし，たとえカードを入れたとしても，1枚のカードがあまりにも専門的なため，結局2枚ほど欲しくなって，やや中途半端で満足に使えないというのが，CM-64ユーザーのホンネでしょう(管弦楽器系とポップスのためのドラムセット，エフェクトのかかったギター数種が入った，まさにCM-64の弱めの音色を補うべきカードが1枚発売されれば，そうともいえないのですが)。

そのCM-64でも，ブラス系はそこそこ入っているので，うまく使えばそれなりに使えます。今回は詳しいデータは引き出せなかったのですが，51版のTP/TRB(トランペット/トロンボーン)は，ゲートタイムをうまく設定し，ヴィロシティをハッキリと変えて使えば，アクセントとして使えそうです。

トランペット(47番)は，聴けばすぐにわかるトランペットの音ですが，これは使うところがない音色といっても過言ではないくらいです。この手の音色はヘタに使うとボロボロになってしまうので，控えたほうが無難かもしれません。

結局，カード6(オーケストラウインド)がないと，まともにアンサンブルが組めないので，CM-64は，単体ではブラスアンサンブルの曲をDTMするのには向いてない楽器かもしれません。

対してSC-55は，とりあえずひととおり音がそろっています。どれも「らしい」音が入っていてそれなりには使えるのですが，1つひとつの音色にいまいち表情がないため，深く突っ込んだ曲を作るにはかなり役不足になってしまいます。

結局，私は金管楽器の曲を作るには，CM-64＋カード6がいちばんいいと思います。しかし，この組み合わせでは，もしポップスにするなら，ドラムがカスになる，ギターがなくなる(いまさらポップスにナチュラルトーンオンリーはないでしょう)，また，フルトゥッティの場合は弦が腐る，などの難点があります。そんな理由からいまいちなのは，残念ながら否定できない事実でしょう。

そう考えてしまうと，シンセでお手軽にそれなりの金管楽器を演奏するのは，まだまだ先の話なのかもしれません。

§おわりに

今回は，ルールというか，蘊蓄に近いテクニックを書いてみました。いまいち何をいってるのかわからないかもしれませんが，ノウハウを言葉にしているのだから，それでいいという気もしたりします。

今回は「和声を理解している」ことを前提に話を進めてしまったので，初心者には(初心者の読者は少ないかもしれませんが)ちょっとわかりにくいかもしれません。

まあ，いまは理解できなくても，知識はあとになって役立つこともあるでしょう。

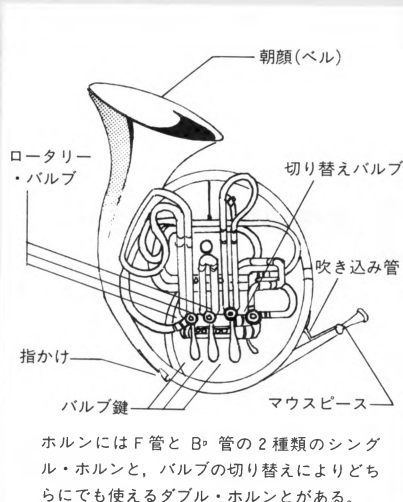
それではまた。

気にとめてなかったのが，その失態を招いた原因ですが，実際に楽器を見てしまえば一目瞭然，これはどう見ても木管楽器。イングリッシュホルンは，オーボエの一種で，いわれてみれば，それに近い音です。ドヴォルザークの交響曲「新世界から」の第2楽章の主題を独奏している楽器ですから，CDでも買ってきて聴いてみれば，どんな音色かわかるでしょう。

ホルンのオーケストラでの役割については，本文を参照してください。

英語	Horn
イタリア語	Corno
フランス語	Cor
ドイツ語	Horn

図A　ダブル・ホルン

ホルンにはF管とB♭管の2種類のシングル・ホルンと，バルブの切り替えによりどちらにでも使えるダブル・ホルンとがある。

●チューバ

テューバとも，トゥバともいいますが，私は「ちうば」とひらがなでいうのがいちばん好きです(間違いですから真似しないように)。

ゴートン・ヤコブ氏の言葉を借りれば，「この，美しく管弦楽における金管楽器の最も深味ある音の楽器」だそうですが，私にいわせれば，「表情の多い，演奏者の性格をよく引き出す楽器」ともなってしまいます(先月私の友人のチューバがすごくえっちだといいましたが，彼に聞いたら「演奏者の性格じゃない？」と教えてくれました)。

チューバはもともと，トロンボーンを除く低音の金管楽器のことを漠然と指す言葉です。古代ローマ時代に軍用に用いられた直管の大きな音が鳴るトランペットの低音楽器「トゥバ」から由来しています。

英語	Tuba
イタリア語	Tuba
フランス語	Tuba
ドイツ語	Tuba

図B　ホルンの音域

F管
実音　記譜

E♭管
実音　記譜

●サクスホルン

本編では扱いませんでしたが，金管楽器としての，サクスホルンというものがあります。

気をつけてほしいのは，これはジャズなどでもよく使われている「サックス」とは違う楽器だということです。あれは「サクソフォン」のことですが，まったく関係がないわけではなく，実は考案者が同じ人なのです。

サクスホルンは，150年ほど前にベルギーのアドルフ・サックスが考案した金管楽器の一属で，弦楽器のヴァイオリン属(ヴァイオリン，ヴィオラ，チェロ，コントラバス)に対して，金管楽器でも統一された音色を作ろうという意図から作られました。全部で7本あり(金管楽器は音域が狭く，和声が密集配置のため)，これらはその調がそのまま名前になっています。しかし，このサクスホルンは現在ではあまり使われていません。

ところでサクソフォンのほうですが，あれは実は金管楽器ではなく，その形状と仕組みから木管楽器に分類されます。管の部分は金属製ですが，クラリネットなどと同様に，リードを振動させて音を出します。

＊　＊　＊

そのほかにも金管楽器には，行進のときに使われるスーザフォン(かのスーザが作ったということはすぐにわかりますよね)などもあり，それなども研究すればまた面白いかもしれませんが，まあ，そこまで深く突っ込んで考える必要のある人はきっと少ないので，このへんで終わりにしておきましょう。

図C　スーザフォン

illustlation:T.Takahashi

マシン語カクテル in Z80's Bar ———第39回

今月は，めくったカードに書かれた数字から爆弾の位置を推理し，8×8のカードから10個の爆弾を探し出す，Windowsでお馴染みのマインスイーパーを作ります。ダンプリストも掲載されているので，あちこちいじくりながら遊んでみてください。

必殺！ 爆弾掃除人(基本編)

Kaneko Shunichi **金子 俊一**

カランコロ～ン♪

源光(以下光)：こんにちは。

ようこ(以下Yo)：あけましておめでとうございます。

光：ってようこさん，もう2月号なのに。振袖着て髪型変えてるなんて，高橋君が困っちゃいますよ。

マスター(以下M)：先月号は丸坊主にされたし。

Yo：似合わないっていいたいの。

光：そんなことはいってませんよ。

M：いやね，ようこちゃんったら正月気分が抜けないんですよ。

光：それって，ボケなんですかね？

M：さあ？

長老(以下老)：おお光か，お年玉をやろう。

光：こりゃだめだ。

M：それとも1月号の原稿を柴田君にまかせたことを間接的に責めているとか。

光：うっ。

老：ほれ光，甘酒でも頼んだらどうじゃ。今日はワシのおごりじゃ。

光：やけに気前がいいなあ。ひょっとして宝くじでも当たったんですか？

老：うむ，たんまり当たっておったぞ。切手シートがな。

Yo：それってお年玉つき年賀ハガキじゃないの？

光&M：おお～っ，ようこちゃんがツッコミを入れた。

爆弾掃除人（基本バージョン）

光：夢でも見てるんだろうか。

M：初春の珍事ってやつですかね。

光：悪いことがおこらなきゃいいけど。

M：最近のボケ方はすごかったからね。

Yo：なにコソコソ話してんの，2人して。

光&M：い，いや別に……。

ゲームを作ろう

Yo：ねえねえ光君，双六でもやろうよ。

老：ワシは百人一首のほうがよいなあ。

Yo：だったらカルタで妥協してあげる。

老：花札でもよいぞ。

光：いいかげん，正月気分から抜けたらどうです。

老：どちらにしても今月はゲームじゃな。

M：どうせだったらコンピュータでやりましょうよ。

光：ぎくっ。

Yo：えーと，マインスイーパーがいいや。

光：マインスイーパーって，あのWindowsとかの？

Yo：そう。

光：渋めのものを選んだね。

Yo：だってプログラム作るの簡単そうじゃない？

光：この展開ってやっぱりいつものパターンにつながるんですよね。

Yo：作ってね，光君。

光：今度から指名料取ろうかな。

マインスイーパーってなに？

老：ワシは知らんぞ。エッグマフィンスイートポテトの略か？

光：なにわけのわからんことをいってるんですか。

M：簡単にいえば爆弾探しですね。

光：そうそう，ルールは簡単。裏返ってるカードをめくっていって，爆弾以外のカードを全部めくれたら終わり。

M：カードっていうか，マス目のほうが正しいと思うけど。

光：カードのほうが感覚的にわかりやすいですよ。

老：なるほど。

Yo：その爆弾をマインっていうのよ。

光：それでマイン（爆弾）スイーパー（掃除人）ですね。

老：確率のゲームなのかのう？

光：違うんですよ。

Yo：えっとね，1個のマス目にはその周りの8個のマス目の中に何個の爆弾があるか書いてあるのよ。

老：？（わかっていない）

光：いくつか簡単なサンプルを書いてみましょう（図1）。図1の“＊”が爆弾で，そのほかが周りに何個爆弾があるか書いてあるカードです。

老：？（まだわからない）

光：しばらく悩んでいてください。

Yo：最初の何個かは運だけど，あとはめくったカードに書かれている爆弾の個数から推理すれば，爆弾の場所がわかるのよ。

光：これは，やってもらったほうがわかりやすいと思いますけどね。

M：それって墓穴を掘るってやつですよ。

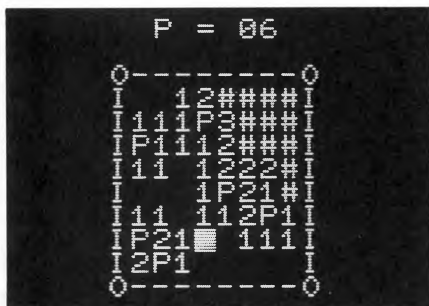
Yo：早く作ってね。

爆弾掃除人基本セット

光：それじゃあいちばん基本的な8×8のマス目に10個の爆弾でいきましょう。

Yo：もちろん時間は計れるんでしょ？

図1　カードのパターン

(1)	(2)	(3)	(4)
＊10	＊2＊	111	＊＊2
110	121	1＊1	23＊
000	000	111	011

光：う～ん。S-OS用だからなあ，ちょっと厳しいかな。
Yo：機種別プログラムでもつけたら？
光：それは読者におまかせするということで。
Yo：しようがないなあ。でもマウスには対応するんでしょ？
光：なに寝言いってんですか。S-OS用なんですよ。
Yo：許してあげるから早く作ってね。
光：それじゃあやりますかね。カチャカチャ……。
Yo：わくわく，わくわく。
老：そろそろできるかな？
光：カチャカチャ……。
M：もうできるでしょ。
光：カチャカチャ……。
老：今日は長いのう。ショートプログラムなんじゃなかったのかのう。
光：できたっと。
Yo：なんだか時間かかったわね。
光：リストを見てくださいよ。
Yo：なんだ，結構長いじゃないの。

アルゴリズムの話

M：光君，解説。
光：あいあいさー，っていってもマス目作って，乱数で爆弾を置いて，あとはキー入力してるだけですよ。
老：なるほど。
Yo：それじゃ解説になってないわよ。
光：ではマス目の作り方からいきますか。
Yo：まってました大統領！
M：ちょっと違うような気がするんだけど。
光：えっと，画面にマス目（カード）を表示させる方法としては，大まかにいって2とおりあると思うんですよ。
Yo：その心は？
光：ひとつは8×8（64バイト）のマス目を用意する方法。もうひとつは10バイト程度の配列を用意する方法です。
Yo：その違いは？
光：前者では爆弾を直接マス目に書いてしまうんですよ。
老：ふむふむ。
光：後者は爆弾の場所だけを配列に保存しておく。
Yo：同じことができるんだったら，だんぜん後者じゃない。バイト数少ないもの。
光：ところが，後者では爆弾の数を表示するときに毎回数えなくちゃならない。たった8つのマス目だからマシン語だったら一瞬で終わるけどね。
老：しかし，あらかじめ計算しておいたほうが早いのう。
光：そのとおりなんですよ。答えを作っておいて表示するだけのほうがよっぽど楽だし，マシンパワーも食わない。
Yo：メモリもたかだか100バイトかそこいらだったら同じようなものね。
光：ええ，ってことで私は前者でプログラミングしました。実際に内部では10×10のマス目を使っているんですけどね。
老：角とかの処理じゃな。
光：するどいですね。
Yo：というと？
光：いちばん外の部分は周りに8マスもないでしょ（図2）。
Yo：確かにはじっこは6マスしかないし，4隅は4マスしかないわね。
光：それをいちいちチェックしてもよかったんですけど，周りにもう1つずつマスがあることにして，同じプログラムで8回計算させてしまったわけですよ。
Yo：なるほどね。

違いはあるの？

Yo：これって，結構似てると思うんだけど，本物と違うの？
光：違いは結構ありますよ。たとえば，
1)　時間の表示がない&残らない
2)　サイズを変えられない
3)　最初からマインを開けてしまうことがある
4)　ヘルプが出ない
ってところかなあ。
老：3つ目はなんじゃ？
光：どうも本物ではひとつ目を開けてから爆弾を置いているのか，爆弾だったらどこかほかのマスとスワッピングをするのかわからないんですけど，とにかくひとつ目から爆弾に当たることはないんですよ。
Yo：経験論ね。
光：対策はあるんですけどね。
老：さっき自分でいったとおりにすればいいんじゃろう。
光：ええ，ちゃんと考えたんですけどね。とりあえずやめておきました。
Yo：ねえ，「ヘルプが出ない」ってことは，本物は「ヘルプが出る」ってことでしょ。手助けしてくれるの？
光：それってジェノサイドで死にそうになったときにHELPキーを押すようなもんですよ。
M：一般的にヘルプっていうと説明とか使い方などのことだよね。
老：ワシでも知っとるぞ。
Yo：なあんだ。ところで時間はやっぱり無理だったのね。
光：拡張可能なようにしておきましたから，「自分でどうぞ」って感じかな。
M：サイズくらい変えられてもよかったんじゃないの？
光：えっと，できる限り拡張しやすいようにしておいたんですよ。
老：これも自分でやれってか。
光：ワークエリアにあるパラメータをいじるだけで大丈夫ですよ。
M：具体的にはどこをいじればいいんですか？
光：えっと，ラベル名でいうのなら，LENGTH,WIDTH,MINES,LOCの4つで

図2　角に置かれた爆弾

```
*1  1*2
23  345
```

爆弾掃除人で遊ぶ

まずは，キー操作の説明。
I,M,J,L：上下左右の移動
Z：めくる
X：P，?，#
G：やり直し（別の面になる）
!：EXIT
Xキーを押すとP，?，#の順に変化するが，それぞれの意味は，
P：爆弾があるという印
?：爆弾があるかもしれないという印
#：初期状態
である。この初期状態“#”でないと“Z”でめくることはできない。これは，プレイ中爆弾を発見したときにつける目印であって，使用しなくてもいい。
8×8のマス目から10個のマインを特定するとクリアになるが，正確には54個のマス目をオープンさせたらクリア。“P”というマークを付けなくてもクリアはできる。また，その周囲にひとつも爆弾がなかった場合はスペースが表示される。このとき，爆弾がないことがわかりきっているところは自動的にコンピュータが開けてくれる。
爆弾を開けてしまうとゲームオーバーになる。“P”を間違えて立てていた場合には，その地点に“×”印が表示される。また，なにかキーを押すとゲームが始まる。
“G”がいつでも有効なのは，タイムゲームがサポートされると便利な機能だからである。最初に何個か開けてみて，よいタイムが出そうもないときは，迷わず“G”という使い方をする。
画面モードは40×25で遊ぶことを前提に作っているので，ゲーム開始前に“W”コマンドで画面モードを切り替えておくように。

大丈夫じゃないですか。それぞれ制限があるけど気をつけていじれば大丈夫。

Yo：たとえば？

光：えっと，横と縦は面積で256以内に収めなければならないとか，表示位置は上を2段分空けておかないといけない。

老：いわゆる処理の都合上ってやつじゃのう。

Yo：2段上には「P」の数が表示されるためとかなのね。

光：そうです。ほかにも爆弾を極端に多く(面積比90％とか)すると，最初の設定にむちゃくちゃ時間がかかるようになってしまう可能性が高い，とか。

老：ただの乱数でマインを置いているから，同じ場所だとまた乱数を作りにいっとるわけじゃな。

光：空いている場所を見つけるまで乱数を作りますからね。

M：まだ，注意点とかあるんですか？

光：えっと，乱数にリフレッシュレジスタをからめてあるんで，機種によっては偏った乱数が発生してしまうかもしれないですね。

老：もちろん，対策あるんじゃろ。

光：ええ，ラベル名だとTEMPというところに初期値が入っていますから，ここを直すといいでしょう。

Yo：えっと，DW $0064ってなっているみたいだけど。

光：上位バイトは常に0にしておいてください。

M：いやあ，それにしてもプログラム大きいから長老におごってもらうまでもありませんね，こりゃ。

老：うむ，めでたいのう。

光：今月もなんだか仕組まれていたような気がする。

Yo：それはいいっこなしなの。

面白いよ

老：え〜と，ここが3でこっちが2じゃからと。

Yo：全部開いたわ。

光：上達が早いですね。

老：これで時間表示やスコア記録ができればのう。

Yo：やっぱりそう思うわよね。

老：うむ，とはいえなかなか面白いではないか。

Yo：でしょ〜。

M：ボケの防止になりますよ，これ。

光：あっ，だからようこちゃんボケなくなったんだ。

M：なるほど！

Yo：失礼しちゃうわね。

—つづく—

リスト1

```
0000                1 ;  MINESWEEPER ?
0000                2 ;
0000                3 ;   by Hikaru Minamoto
0000                4
0000                5
C000                6         ORG     $C000
C000                7
C000                8 #PRTHX  EQU     $1FC1
C000                9 #MPRINT EQU     $1FE2
C000               10 #MSX    EQU     $1FE5
C000               11 #PRINT  EQU     $1FF4
C000               12 #CSR    EQU     $2018
C000               13 #LOC    EQU     $201E
C000               14 #FLGET  EQU     $2021
C000               15
C000               16 ;       Main Routine
C000               17
C000               18 COLD
C000 CD B5 C6      19         CALL    TIME_LOAD ; Reserve
C003               20 START
C003 CD 24 C0      21         CALL    CLS
C006 CD DB C2      22         CALL    INI_DAT
C009 CD 14 C2      23         CALL    MAKE
C00C CD 6F C3      24         CALL    WAKU
C00F 2A 25 C4      25         LD      HL,(LOC2)
C012 CD 1E 20      26         CALL    #LOC
C015               27 LOOP    ;
C015 11 71 C5      28         LD      DE,DATA2
C018 CD DB C3      29         CALL    PR_MAP
C01B CD 2A C0      30         CALL    KEYIN
C01E CD B5 C6      31         CALL    TIME_PRINT ; Reserve
C021 C3 15 C0      32         JP      LOOP
C024               33
C024               34 ;       Sub Routine
C024               35
C024               36 CLS;*   in = nothing
C024 3E 0C         37         LD      A,$0C
C026 CD F4 1F      38         CALL    #PRINT
C029 C9            39         RET
C02A               40
C02A               41 KEYIN;*
C02A CD 18 20      42         CALL    #CSR
C02D CD 21 20      43         CALL    #FLGET
C030 FE 5A 20      44         IF A="Z" THEN JR OPEN
C033 02 18 53
C036 FE 58 20      45         IF A="X" THEN JP CHR_SET
C039 03 C3 98
C03C C1
C03D FE 4F 20      46         IF A="O" THEN JP OPTION     ; Reserve
C040 03 C3 B6
C043 C6
C044 FE 54 20      47         IF A="T" THEN JP TIME_SCORE ; Reserve
C047 03 C3 B5
C04A C6
C04B FE 21 20      48         IF A="!" THEN JR QUIT
C04E 02 18 33
C051 FE 4C 20      49         IF A="L" THEN INC L
C054 01 2C
C056 FE 4A 20      50         IF A="J" THEN DEC L
C059 01 2D
C05B FE 49 20      51         IF A="I" THEN DEC H
C05E 01 25
C060 FE 4D 20      52         IF A="M" THEN INC H
C063 01 24
C065 FE 47 20      53         IF A="G" THEN JP MORE
C068 03 C3 7F
C06B C1
C06C               54         ;
C06C 3A 1E C4      55         LD      A,(LOC+1)
C06F BC            56         CP      H
C070 D0            57         RET     NC
C071 3A 2C C4      58         LD      A,(LOC5+1)
C074 BC            59         CP      H
C075 D8            60         RET     C
C076 3A 1D C4      61         LD      A,(LOC)
C079 BD            62         CP      L
C07A D0            63         RET     NC
C07B 3A 2B C4      64         LD      A,(LOC5)
C07E BD            65         CP      L
C07F D8            66         RET     C
C080 CD 1E 20      67         CALL    #LOC
C083 C9            68         RET
C084               69 QUIT
C084 F1            70         POP     AF
C085 CD B5 C6      71         CALL    TIME_SAVE ; Reserve
C088 C9            72         RET
C089               73 OPEN
C089 CD FC C1      74         CALL    GET_ADR
C08C 21 71 C5      75         LD      HL,DATA2
C08F CD 79 C2      76         CALL    PEEK
C092 57            77         LD      D,A
C093 3A 0F C4      78         LD      A,(KABE)
C096 BB            79         CP      E
C097 C0            80         RET     NZ
C098 7A            81         LD      A,D
C099 21 2D C4      82         LD      HL,DATA1
C09C CD 79 C2      83         CALL    PEEK
C09F CD C3 C0      84         CALL    OPEN2
C0A2 21 71 C5      85         LD      HL,DATA2
C0A5 CD 74 C2      86         CALL    POKE
C0A8               87         ;
C0A8 3A 10 C4      88         LD      A,(MINE)
C0AB BB            89         CP      E
C0AC CA 46 C1      90         JP      Z,FALSE
C0AF               91         ;
C0AF 3A FD C3      92         LD      A,(@OPEN)
C0B2 3C            93         INC     A
C0B3 32 FD C3      94         LD      (@OPEN),A
C0B6 E5            95         PUSH    HL
C0B7 21 22 C4      96         LD      HL,SUM_MINE
C0BA BE            97         CP      (HL)
C0BB E1            98         POP     HL
C0BC C0            99         RET     NZ
C0BD 11 FF C3     100         LD      DE,DONE
C0C0 C3 49 C1     101         JP      FUL_OPN
C0C3              102 OPEN2
C0C3 7B           103         LD      A,E
C0C4 FE 30        104         CP      "0"
C0C6 7A           105         LD      A,D
C0C7 C0           106         RET     NZ
C0C8 21 71 C5     107         LD      HL,DATA2
C0CB CD 74 C2     108         CALL    POKE
C0CE              109         ;
C0CE 21 71 C5     110         LD      HL,DATA2
C0D1 CD 7E C2     111         CALL    CUL_ADR
C0D4 DD 21 FF     112         LD      IX,$FFFF
C0D7 FF
C0D8 DD E5        113         PUSH    IX ; END_MARK
C0DA E5           114         PUSH    HL ; START_POINT
C0DB              115         ;
```

```
C0DB                116 SPC_CHK
C0DB E1             117           POP     HL
C0DC 23             118           INC     HL
C0DD 7C             119           LD      A,H
C0DE B5             120           OR      L
C0DF 7A             121           LD      A,D
C0E0 1E 20          122           LD      E," "
C0E2 C8             123           RET     Z  ;HL=FFFF
C0E3                124           ;
C0E3 3A 20 C4       125           LD      A,(WID2)
C0E6 06 00          126           LD      B,0
C0E8 4F             127           LD      C,A
C0E9 09             128           ADD     HL,BC
C0EA CD 1D C1       129           CALL    SPC_CHK2 ;9
C0ED 2B             130           DEC     HL
C0EE CD 1D C1       131           CALL    SPC_CHK2 ;8
C0F1 2B             132           DEC     HL
C0F2 CD 1D C1       133           CALL    SPC_CHK2 ;7
C0F5 3A 1B C4       134           LD      A,(WIDTH)
C0F8 06 00          135           LD      B,0
C0FA 4F             136           LD      C,A
C0FB B7             137           OR      A
C0FC ED 42          138           SBC     HL,BC
C0FE CD 1D C1       139           CALL    SPC_CHK2 ;6
C101 2B             140           DEC     HL
C102 2B             141           DEC     HL
C103 CD 1D C1       142           CALL    SPC_CHK2 ;4
C106 B7             143           OR      A
C107 3A 1B C4       144           LD      A,(WIDTH)
C10A 06 00          145           LD      B,0
C10C 4F             146           LD      C,A
C10D ED 42          147           SBC     HL,BC
C10F CD 1D C1       148           CALL    SPC_CHK2 ;3
C112 2B             149           DEC     HL
C113 CD 1D C1       150           CALL    SPC_CHK2 ;2
C116 2B             151           DEC     HL
C117 CD 1D C1       152           CALL    SPC_CHK2 ;1
C11A C3 DB C0       153           JP      SPC_CHK
C11D                154 SPC_CHK2
C11D 3A 0F C4       155           LD      A,(KABE)
C120 BE             156           CP      (HL)
C121 C0             157           RET     NZ
C122 E5             158           PUSH    HL
C123 01 71 C5       159           LD      BC,DATA2
C126 B7             160           OR      A
C127 ED 42          161           SBC     HL,BC
C129 01 2D C4       162           LD      BC,DATA1
C12C 09             163           ADD     HL,BC
C12D 7E             164           LD      A,(HL)
C12E FE 30          165           CP      "0"
C130 20 02          166           JR      NZ,SPC_CHK3
C132 3E 20          167           LD      A," "
C134                168 SPC_CHK3
C134 E1             169           POP     HL
C135 77             170           LD      (HL),A
C136 08             171           EX      AF,AF'
C137 3A FD C3       172           LD      A,(@OPEN)
C13A 3C             173           INC     A
C13B 32 FD C3       174           LD      (@OPEN),A
C13E 08             175           EX      AF,AF'
C13F FE 20          176           CP      " "
C141 C0             177           RET     NZ
C142 C1             178           POP     BC
C143 E5             179           PUSH    HL
C144 C5             180           PUSH    BC
C145 C9             181           RET
C146                182 FALSE
C146 11 06 C4       183           LD      DE,FAIL
C149                184 FUL_OPN
C149 2A 27 C4       185           LD      HL,(LOC3)
C14C CD 1E 20       186           CALL    #LOC
C14F CD E5 1F       187           CALL    #MSX
C152 3A 21 C4       188           LD      A,(SUM)
C155 47             189           LD      B,A
C156                190 FUL_OPN2
C156 48             191           LD      C,B
C157 0D             192           DEC     C
C158 79             193           LD      A,C
C159 21 2D C4       194           LD      HL,DATA1
C15C CD 79 C2       195           CALL    PEEK
C15F 53             196           LD      D,E
C160 3E 30          197           LD      A,"0"
C162 BB             198           CP      E
C163 CC 8E C1       199           CALL    Z,PUT_SPC
C166                200           ;
C166 79             201           LD      A,C
C167 21 71 C5       202           LD      HL,DATA2
C16A CD 79 C2       203           CALL    PEEK
C16D 3A 0D C4       204           LD      A,(POLE)
C170 BB             205           CP      E
C171 CC 83 C1       206           CALL    Z,FUL_OPN3
C174 10 E0          207           DJNZ    FUL_OPN2
C176                208
C176 11 2D C4       209           LD      DE,DATA1
C179 CD DB C3       210           CALL    PR_MAP
C17C CD 21 20       211           CALL    #FLGET
C17F                212 MORE
C17F F1             213           POP     AF
C180 C3 03 C0       214           JP      START
C183                215 FUL_OPN3
C183 3A 10 C4       216           LD      A,(MINE)
C186 BA             217           CP      D
C187 C8             218           RET     Z
C188 3A 11 C4       219           LD      A,(HAZU)
C18B 5F             220           LD      E,A
C18C 18 02          221           JR      PUT_SPC+2
C18E                222 PUT_SPC
C18E 1E 20          223           LD      E," "
C190 79             224           LD      A,C
C191 21 2D C4       225           LD      HL,DATA1
C194 CD 74 C2       226           CALL    POKE
C197 C9             227           RET
C198                228           ;
C198                229 CHR_SET
C198 CD FC C1       230           CALL    GET_ADR ; A=0-63
C19B 21 71 C5       231           LD      HL,DATA2
C19E CD 79 C2       232           CALL    PEEK
C1A1 57             233           LD      D,A
C1A2 7B             234           LD      A,E
C1A3 21 0D C4       235           LD      HL,POLE ; POLE
C1A6 BE 20 02       236           IF A=(HL) THEN JR SET_CHR
C1A9 18 0D
C1AB 23             237           INC     HL      ; MARK
C1AC BE 20 02       238           IF A=(HL) THEN JR SET_CHR
C1AF 18 07
C1B1 23             239           INC     HL      ; KABE
C1B2 BE 20 02       240           IF A=(HL) THEN JR SET_POLE
C1B5 18 1C
C1B7 C9             241           RET
C1B8                242 SET_CHR
C1B8 23             243           INC     HL
C1B9 5E             244           LD      E,(HL)
C1BA 7A             245           LD      A,D
C1BB 21 71 C5       246           LD      HL,DATA2
C1BE CD 74 C2       247           CALL    POKE
C1C1 3A 0F C4       248           LD      A,(KABE)
C1C4 BB             249           CP      E
C1C5 C8             250           RET     Z
C1C6 3A FE C3       251           LD      A,(@PNUM)
C1C9 B7             252           OR      A
C1CA 3D             253           DEC     A
C1CB 27             254           DAA
C1CC 32 FE C3       255           LD      (@PNUM),A
C1CF CD EA C1       256           CALL    PR_PNUM
C1D2 C9             257           RET
C1D3                258 SET_POLE
C1D3 2B             259           DEC     HL
C1D4 2B             260           DEC     HL
C1D5 5E             261           LD      E,(HL)
C1D6 7A             262           LD      A,D
C1D7 21 71 C5       263           LD      HL,DATA2
C1DA CD 74 C2       264           CALL    POKE
C1DD                265           ;
C1DD 3A FE C3       266           LD      A,(@PNUM)
C1E0 B7             267           OR      A
C1E1 3C             268           INC     A
C1E2 27             269           DAA
C1E3 32 FE C3       270           LD      (@PNUM),A
C1E6 CD EA C1       271           CALL    PR_PNUM
C1E9 C9             272           RET
C1EA                273 PR_PNUM
C1EA CD 18 20       274           CALL    #CSR
C1ED E5             275           PUSH    HL
C1EE 2A 29 C4       276           LD      HL,(LOC4)
C1F1 CD 1E 20       277           CALL    #LOC
C1F4 CD C1 1F       278           CALL    #PRTHX
C1F7 E1             279           POP     HL
C1F8 CD 1E 20       280           CALL    #LOC
C1FB C9             281           RET
C1FC                282 GET_ADR           ; in = HL ,out = A (0-63)
C1FC C5             283           PUSH    BC
C1FD ED 5B 25       284           LD      DE,(LOC2)
C200 C4
C201 B7             285           OR      A
C202 ED 52          286           SBC     HL,DE
C204 7C             287           LD      A,H
C205 B7             288           OR      A
C206 28 09          289           JR      Z,GET_ADR3
C208 3A 1B C4       290           LD      A,(WIDTH)
C20B 47             291           LD      B,A
C20C AF             292           XOR     A
C20D                293 GET_ADR2
C20D 80             294           ADD     A,B
C20E 25             295           DEC     H
C20F 20 FC          296           JR      NZ,GET_ADR2
C211                297 GET_ADR3
C211 85             298           ADD     A,L
C212 C1             299           POP     BC
C213 C9             300           RET
C214                301
C214                302 MAKE;*
C214 3A 1C C4       303           LD      A,(MINES)
C217 4F             304           LD      C,A
C218 3A 10 C4       305           LD      A,(MINE)
C21B 47             306           LD      B,A
C21C                307 MAKE2
C21C CD A0 C2       308           CALL    RND
C21F 57             309           LD      D,A
C220 3A 21 C4       310           LD      A,(SUM)
C223 3D             311           DEC     A
C224 BA             312           CP      D
C225 38 F5          313           JR      C,MAKE2
C227 7A             314           LD      A,D
C228 21 2D C4       315           LD      HL,DATA1
C22B CD 79 C2       316           CALL    PEEK
C22E 78             317           LD      A,B
C22F BB             318           CP      E
C230 28 EA          319           JR      Z,MAKE2
C232 7A             320           LD      A,D
C233 58             321           LD      E,B
C234 21 2D C4       322           LD      HL,DATA1
C237 CD 74 C2       323           CALL    POKE
C23A CD 41 C2       324           CALL    HYOKA
C23D 0D             325           DEC     C
C23E 20 DC          326           JR      NZ,MAKE2
C240 C9             327           RET
C241                328 HYOKA
C241 C5             329           PUSH    BC
C242 2B             330           DEC     HL
C243 CD 6E C2       331           CALL    DAT_INC ;4
C246 23             332           INC     HL
C247 23             333           INC     HL
C248 CD 6E C2       334           CALL    DAT_INC ;6
C24B 3A 20 C4       335           LD      A,(WID2)
C24E 06 00          336           LD      B,0
C250 4F             337           LD      C,A
C251 B7             338           OR      A
C252 ED 42          339           SBC     HL,BC
C254 CD 6E C2       340           CALL    DAT_INC ;9
C257 2B             341           DEC     HL
C258 CD 6E C2       342           CALL    DAT_INC ;8
C25B 2B             343           DEC     HL
C25C CD 6E C2       344           CALL    DAT_INC ;7
C25F 09             345           ADD     HL,BC
C260 09             346           ADD     HL,BC
C261 CD 6E C2       347           CALL    DAT_INC ;1
C264 23             348           INC     HL
C265 CD 6E C2       349           CALL    DAT_INC ;2
C268 23             350           INC     HL
C269 CD 6E C2       351           CALL    DAT_INC ;3
C26C C1             352           POP     BC
C26D C9             353           RET
C26E                354 DAT_INC
C26E 7E             355           LD      A,(HL)
C26F BB             356           CP      E
C270 C8             357           RET     Z
C271 3C             358           INC     A
C272 77             359           LD      (HL),A
```

```
C273 C9          360          RET
C274             361
C274             362 POKE;*   in = HL (DATA1 or DATA2) ,A<64 ,E=DATA
C274 CD 7E C2    363          CALL    CUL_ADR
C277 73          364          LD      (HL),E
C278 C9          365          RET
C279             366 PEEK;*   in = HL (DATA1 or DATA2) ,A<64 ,out = E
C279 CD 7E C2    367          CALL    CUL_ADR
C27C 5E          368          LD      E,(HL)
C27D C9          369          RET
C27E             370 CUL_ADR
C27E F5          371          PUSH    AF
C27F D5          372          PUSH    DE
C280 08          373          EX      AF,AF'
C281 3A 1B C4    374          LD      A,(WIDTH)
C284 5F          375          LD      E,A
C285 16 00       376          LD      D,0
C287 08          377          EX      AF,AF'
C288             378 CUL2
C288 14          379          INC     D
C289 93          380          SUB     E
C28A 30 FC       381          JR      NC,CUL2
C28C 83          382          ADD     A,E
C28D F5          383          PUSH    AF
C28E 3A 20 C4    384          LD      A,(WID2)
C291 5F          385          LD      E,A
C292 7A          386          LD      A,D
C293 16 00       387          LD      D,0
C295             388 CUL3
C295 19          389          ADD     HL,DE
C296 3D          390          DEC     A
C297 20 FC       391          JR      NZ,CUL3
C299 F1          392          POP     AF
C29A 5F          393          LD      E,A
C29B 19          394          ADD     HL,DE
C29C 23          395          INC     HL
C29D D1          396          POP     DE
C29E F1          397          POP     AF
C29F C9          398          RET
C2A0             399
C2A0             400 RND;*    out = A
C2A0 C5          401          PUSH    BC
C2A1 D5          402          PUSH    DE
C2A2 E5          403          PUSH    HL
C2A3 2A 18 C4    404          LD      HL,(TEMP)
C2A6 ED 5F       405          LD      A,R
C2A8 AC          406          XOR     H
C2A9 67          407          LD      H,A
C2AA AD          408          XOR     L
C2AB 6F          409          LD      L,A
C2AC 54 5D       410          LD      DE,HL
C2AE CD C9 C2    411          CALL    MULTI
C2B1 CB 3C       412          SRL     H
C2B3 CB 1D       413          RR      L
C2B5 CB 3C       414          SRL     H
C2B7 CB 1D       415          RR      L
C2B9 CB 3C       416          SRL     H
C2BB CB 1D       417          RR      L
C2BD CB 3C       418          SRL     H
C2BF CB 1D       419          RR      L
C2C1 7D          420          LD      A,L
C2C2 32 18 C4    421          LD      (TEMP),A
C2C5 E1          422          POP     HL
C2C6 D1          423          POP     DE
C2C7 C1          424          POP     BC
C2C8 C9          425          RET
C2C9             426 MULTI
C2C9 3E 10       427          LD      A,16
C2CB 44 4D       428          LD      BC,HL
C2CD 21 00 00    429          LD      HL,$0000
C2D0             430 MUL2
C2D0 29          431          ADD     HL,HL
C2D1 EB          432          EX      DE,HL
C2D2 29          433          ADD     HL,HL
C2D3 EB          434          EX      DE,HL
C2D4 30 01       435          JR      NC,MUL3
C2D6 09          436          ADD     HL,BC
C2D7             437 MUL3
C2D7 3D          438          DEC     A
C2D8 20 F6       439          JR      NZ,MUL2
C2DA C9          440          RET
C2DB             441
C2DB             442 INI_DAT;*
C2DB AF          443          XOR     A
C2DC 32 FD C3    444          LD      (@OPEN),A
C2DF 32 FE C3    445          LD      (@PNUM),A
C2E2             446          ;
C2E2 3A 1A C4    447          LD      A,(LENGTH)
C2E5 47          448          LD      B,A
C2E6 3C          449          INC     A
C2E7 3C          450          INC     A
C2E8 32 1F C4    451          LD      (LEN2),A
C2EB             452          ;
C2EB 3A 1B C4    453          LD      A,(WIDTH)
C2EE 5F          454          LD      E,A
C2EF 3C          455          INC     A
C2F0 3C          456          INC     A
C2F1 32 20 C4    457          LD      (WID2),A
C2F4             458          ;
C2F4 AF          459          XOR     A
C2F5             460 INI2
C2F5 83          461          ADD     A,E
C2F6 10 FD       462          DJNZ    INI2
C2F8 32 21 C4    463          LD      (SUM),A
C2FB             464          ;
C2FB 21 1C C4    465          LD      HL,MINES
C2FE 5E          466          LD      E,(HL)
C2FF 93          467          SUB     E
C300 32 22 C4    468          LD      (SUM_MINE),A
C303             469          ;
C303 3A 1F C4    470          LD      A,(LEN2)
C306 47          471          LD      B,A
C307 05          472          DEC     B
C308 3A 20 C4    473          LD      A,(WID2)
C30B 5F          474          LD      E,A
C30C 16 00       475          LD      D,0
C30E 21 00 00    476          LD      HL,0
C311             477 INI3
C311 19          478          ADD     HL,DE
C312 10 FD       479          DJNZ    INI3
C314 22 23 C4    480          LD      (SUM_WID),HL
C317             481          ;
C317 2A 1D C4    482          LD      HL,(LOC)
C31A 24          483          INC     H
C31B 2C          484          INC     L
C31C 22 25 C4    485          LD      (LOC2),HL
C31F             486          ;
C31F 25          487          DEC     H
C320 25          488          DEC     H
C321 25          489          DEC     H
C322 2D          490          DEC     L
C323 3A 1B C4    491          LD      A,(WIDTH)
C326 CB 3F       492          SRL     A
C328 85          493          ADD     A,L
C329 6F          494          LD      L,A
C32A 2C          495          INC     L
C32B 2C          496          INC     L
C32C 22 29 C4    497          LD      (LOC4),HL
C32F 3D          498          DEC     A
C330 3D          499          DEC     A
C331 6F          500          LD      L,A
C332 22 27 C4    501          LD      (LOC3),HL
C335             502          ;
C335 2A 1D C4    503          LD      HL,(LOC)    ; H=X,L=Y
C338 ED 5B 1A    504          LD      DE,(LENGTH); D=L,E=W
C33B C4
C33C 7C          505          LD      A,H
C33D 83          506          ADD     A,E
C33E 67          507          LD      H,A
C33F 7D          508          LD      A,L
C340 82          509          ADD     A,D
C341 6F          510          LD      L,A
C342 22 2B C4    511          LD      (LOC5),HL
C345             512          ;
C345 3E 30       513          LD      A,"0"
C347 21 2D C4    514          LD      HL,DATA1
C34A 11 2E C4    515          LD      DE,DATA1+1
C34D 01 43 01    516          LD      BC,18*18-1
C350 77          517          LD      (HL),A
C351 ED B0       518          LDIR
C353             519          ;
C353 3A 0F C4    520          LD      A,(KABE)
C356 13          521          INC     DE
C357 23          522          INC     HL
C358 01 43 01    523          LD      BC,18*18-1
C35B 77          524          LD      (HL),A
C35C ED B0       525          LDIR
C35E             526          ;
C35E 2A 27 C4    527          LD      HL,(LOC3)
C361 CD 1E 20    528          CALL    #LOC
C364 CD E2 1F    529          CALL    #MPRINT
C367 50 20 3D    530          DM      "P = 00"
C36A 20 30 30
C36D 00          531          DS      1
C36E C9          532          RET
C36F             533
C36F             534 WAKU;*
C36F 21 2D C4    535          LD      HL,DATA1
C372 CD 78 C3    536          CALL    WAKU1
C375 21 71 C5    537          LD      HL,DATA2
C378             538 WAKU1
C378 E5          539          PUSH    HL
C379 3A 14 C4    540          LD      A,(TOLE)
C37C 77          541          LD      (HL),A
C37D             542          ;
C37D 3A 1B C4    543          LD      A,(WIDTH)
C380 3C          544          INC     A
C381 16 00       545          LD      D,0
C383 5F          546          LD      E,A
C384 3A 15 C4    547          LD      A,(TORI)
C387 19          548          ADD     HL,DE
C388 77          549          LD      (HL),A
C389             550          ;
C389 E1          551          POP     HL
C38A E5          552          PUSH    HL
C38B ED 4B 23    553          LD      BC,(SUM_WID)
C38E C4
C38F 3A 16 C4    554          LD      A,(BOLE)
C392 09          555          ADD     HL,BC
C393 77          556          LD      (HL),A
C394             557          ;
C394 3A 17 C4    558          LD      A,(BORI)
C397 19          559          ADD     HL,DE
C398 77          560          LD      (HL),A
C399             561          ;
C399 3A 13 C4    562          LD      A,(YOKO)
C39C E1          563          POP     HL
C39D E5          564          PUSH    HL
C39E 23          565          INC     HL
C39F 11 01 00    566          LD      DE,$0001
C3A2 CD D0 C3    567          CALL    WAKU3
C3A5 E1          568          POP     HL
C3A6 E5          569          PUSH    HL
C3A7 ED 4B 23    570          LD      BC,(SUM_WID)
C3AA C4
C3AB 03          571          INC     BC
C3AC 09          572          ADD     HL,BC
C3AD CD D0 C3    573          CALL    WAKU3
C3B0             574          ;
C3B0 3A 12 C4    575          LD      A,(TATE)
C3B3 E1          576          POP     HL
C3B4 E5          577          PUSH    HL
C3B5 08          578          EX      AF,AF'
C3B6 3A 20 C4    579          LD      A,(WID2)
C3B9 4F          580          LD      C,A
C3BA 06 00       581          LD      B,0
C3BC 08          582          EX      AF,AF'
C3BD 09          583          ADD     HL,BC
C3BE 50 59       584          LD      DE,BC
C3C0 CD C8 C3    585          CALL    WAKU2
C3C3 E1          586          POP     HL
C3C4 CB 21       587          SLA     C
C3C6 0D          588          DEC     C
C3C7 09          589          ADD     HL,BC
C3C8             590 WAKU2
C3C8 08          591          EX      AF,AF'
C3C9 3A 1A C4    592          LD      A,(LENGTH)
C3CC 47          593          LD      B,A
C3CD 08          594          EX      AF,AF'
C3CE 18 06       595          JR      WAKU4
C3D0             596 WAKU3
C3D0 08          597          EX      AF,AF'
C3D1 3A 1B C4    598          LD      A,(WIDTH)
C3D4 47          599          LD      B,A
C3D5 08          600          EX      AF,AF'
C3D6             601 WAKU4
C3D6 77          602          LD      (HL),A
C3D7 19          603          ADD     HL,DE
```


▶友人のパソコン通信の環境を見て，自分でもやりたくなってしまった今日この頃。モデムとハードディスクがほしいよお。でもはまりそうで怖い。　　川瀬　禎明(20)埼玉県

```
C3D8 10 FC      604             DJNZ     WAKU4
C3DA C9         605             RET
C3DB            606
C3DB            607 PR_MAP;*
C3DB CD 18 20   608             CALL     #CSR
C3DE E5         609             PUSH     HL
C3DF 3A 1F C4   610             LD       A,(LEN2)
C3E2 4F         611             LD       C,A
C3E3 2A 1D C4   612             LD       HL,(LOC)
C3E6            613 PR_MAP0
C3E6 CD 1E 20   614             CALL     #LOC
C3E9 3A 20 C4   615             LD       A,(WID2)
C3EC 47         616             LD       B,A
C3ED            617 PR_MAP1
C3ED 1A         618             LD       A,(DE)
C3EE CD F4 1F   619             CALL     #PRINT
C3F1 13         620             INC      DE
C3F2 10 F9      621             DJNZ     PR_MAP1
C3F4 24         622             INC      H
C3F5 0D         623             DEC      C
C3F6 20 EE      624             JR       NZ,PR_MAP0
C3F8            625             ;
C3F8 E1         626             POP      HL
C3F9 CD 1E 20   627             CALL     #LOC
C3FC C9         628             RET
C3FD            629
C3FD            630 ;           Work & Data Area
C3FD            631
C3FD 00         632 @OPEN       DS       1
C3FE 00         633 @PNUM       DS       1
C3FF            634             ;
C3FF 44 4F 4E   635 DONE        DM       "DONE !"
C402 45 20 21
C405 00         636             DS       1
C406 46 41 4C   637 FAIL        DM       "FALE !"
C409 45 20 21
C40C 00         638             DS       1
C40D            639             ;
C40D 50         640 POLE        DB       "P"  ;P
C40E 3F         641 MARK        DB       "?"  ;?
C40F 23         642 KABE        DB       "#"  ;#
C410 2A         643 MINE        DB       "*"  ;*
C411 58         644 HAZU        DB       "X"  ;X
C412            645             ;
C412 49         646 TATE        DB       "I"  ;I
C413 2D         647 YOKO        DB       "-"  ;-
C414 4F         648 TOLE        DB       "O"  ;O
C415 4F         649 TORI        DB       "O"  ;O
C416 4F         650 BOLE        DB       "O"  ;O
C417 4F         651 BORI        DB       "O"  ;O
C418            652             ;
C418            653 TEMP
C418 64 00      654             DW       $0064; PRIMARY DATA
C41A            655             ;
C41A            656
C41A            657 ;           Size change data
C41A            658 LENGTH
C41A 08         659             DB       8
C41B            660 WIDTH
C41B 08         661             DB       8
C41C            662 MINES
C41C 0A         663             DB       10
C41D            664 LOC
C41D 0F 05      665             DW       $050F
C41F            666
C41F            667 ;           Size work
C41F            668 LEN2                 ; LENGTH+2
C41F 00         669             DS       1
C420            670 WID2                 ; WIDTH+2
C420 00         671             DS       1
C421            672 SUM                  ; LENGTH*WIDTH
C421 00         673             DS       1
C422            674 SUM_MINE             ; SUM-MINE
C422 00         675             DS       1
C423            676 SUM_WID              ; WID2*(LEN2-1)
C423 00 00      677             DS       2
C425            678 LOC2                 ; LOC(+1,+1)
C425 00 00      679             DS       2
C427            680 LOC3                 ; MESSAGE_LOC
C427 00 00      681             DS       2
C429            682 LOC4                 ; P_NUM_LOC
C429 00 00      683             DS       2
C42B            684 LOC5                 ; CURSOLE_MOVE
C42B 00 00      685             DS       2
C42D            686
C42D            687 DATA1
C42D 00 00 00   688             DS       18*18
C571            689 DATA2
C571 00 00 00   690             DS       18*18
C6B5            691
C6B5            692 ;           Reserve
C6B5            693
C6B5            694 TIME_LOAD
C6B5            695 TIME_PRINT
C6B5            696 TIME_SCORE
C6B5            697 TIME_SAVE
C6B5 C9         698             RET
C6B6            699 OPTION
C6B6 C9         700             RET
OBJECT CODE END C6B6
```

リスト2

```
C000 CD B5 C6 0D 24 C0 CD DB : A1
C008 C2 CD 14 C2 CD 6F C3 2A : 8E
C010 25 C4 CD 1E 20 11 71 C5 : 3B
C018 CD DB C3 CD 2A C0 CD B5 : A4
C020 C6 C3 15 C0 3E 0C CD F4 : 69
C028 1F C9 CD 18 20 CD 21 20 : FB
C030 FE 5A 20 02 18 53 FE 58 : 3B
C038 20 03 C3 98 C1 FE 4F 20 : AC
C040 03 C3 B6 C6 FE 54 20 03 : B7
C048 C3 B5 C6 FE 21 20 02 18 : 97
C050 33 FE 4C 20 01 2C FE 4A : 12
C058 20 01 2D FE 49 20 01 25 : DB
C060 FE 4D 20 01 24 FE 47 20 : F5
C068 03 C3 7F C1 3A 1E C4 BC : DE
C070 D0 3A 2C C4 BC D8 3A 1D : E5
C078 C4 BD D0 3A 2B C4 BD D8 : 0F
---------------------------------
SUM: 32 88 BF 8E 20 A2 2C 66 087A

C080 CD 1E 20 C9 F1 CD B5 C6 : 0D
C088 C9 CD FC C1 21 71 C5 CD : 77
C090 79 C2 57 3A 0F C4 BB C0 : 1A
C098 7A 21 2D C4 CD 79 C2 CD : 61
C0A0 C3 C0 21 71 C5 CD 74 C2 : DD
C0A8 3A 10 C4 BB CA 46 C1 3A : D4
C0B0 FD C3 3C 32 FD C3 E5 21 : F4
C0B8 22 C4 BE E1 C0 11 FF C3 : 18
C0C0 C3 49 C1 7B FE 30 7A C0 : B0
C0C8 21 71 C5 CD 74 C2 21 71 : EC
C0D0 C5 CD 7E C2 DD 21 FF FF : CE
C0D8 DD E5 E5 E1 23 7C B5 7A : 56
C0E0 1E 20 C8 3A 20 C4 06 00 : 2A
C0E8 4F 09 CD 1D C1 2B CD 1D : 18
C0F0 C1 2B CD 1D C1 3A 1B C4 : B0
C0F8 06 00 4F B7 ED 42 CD 1D : 25
---------------------------------
SUM: 5F E5 19 DD 3B 5C 1A A8 A6EF

C100 C1 2B 2B CD 1D C1 B7 3A : B3
C108 1B C4 06 00 4F ED 42 CD : 30
C110 1D C1 2B CD 1D C1 2B CD : AC
C118 1D C1 C3 DB C0 3A 0F C4 : 49
C120 BE C0 E5 01 71 C5 B7 ED : 3E
C128 42 01 2D C4 09 7E FE 30 : E9
C130 20 02 3E 20 E1 77 08 3A : 1A
C138 FD C3 3C 32 FD C3 08 FE : F4
C140 20 C0 C1 E5 C5 C9 11 06 : 2B
C148 C4 2A 27 C4 CD 1E 20 CD : B1
C150 E5 1F 3A 21 C4 47 48 0D : BF
C158 79 21 2D C4 CD 79 C2 53 : E6
C160 3E 30 BB CC 8E C1 79 21 : DE
C168 71 C5 CD 79 C2 3A 0D C4 : 49
C170 BB CC 83 C1 10 E0 11 2D : F9
C178 C4 CD DB C3 CD 21 20 F1 : 2E
---------------------------------
SUM: A3 AF E0 E3 F1 C9 EA 23 14DD

C180 C3 03 C0 3A 10 C4 BA C8 : 16
C188 3A 11 C4 5F 18 02 1E 20 : C6
C190 79 21 2D C4 CD 74 C2 C9 : 57
C198 CD FC C1 21 71 C5 CD 79 : 27
C1A0 C2 57 7B 21 0D C4 BE 20 : 64
C1A8 02 18 0D 23 BE 20 02 18 : 42
C1B0 07 23 BE 20 02 18 1C C9 : 07
C1B8 23 5E 7A 21 71 C5 CD 74 : 93
C1C0 C2 3A 0F C4 BB C8 3A FE : 8A
C1C8 C3 B7 3D 27 32 FE C3 CD : 9E
C1D0 EA C1 C9 2B 2B 5E 7A 21 : C3
C1D8 71 C5 CD 74 C2 3A FE C3 : 34
C1E0 B7 3C 27 32 FE C3 CD EA : C4
C1E8 C1 C9 CD 18 20 E5 2A 29 : C7
C1F0 C4 CD 1E 20 CD C1 1F E1 : 5D
C1F8 CD 1E 20 C9 C5 ED 5B 25 : 06
---------------------------------
SUM: 1A 88 46 C0 2E 74 F6 67 15BD

C200 C4 B7 ED 52 7C B7 28 09 : 1E
C208 3A 1B C4 47 AF 80 25 20 : D4
C210 FC 85 C1 C9 3A 1C C4 4F : 74
C218 3A 10 C4 47 CD A0 C2 57 : DB
C220 3A 21 C4 3D BA 38 F5 7A : BD
C228 21 2D C4 CD 79 C2 78 BB : 4D
C230 28 EA 7A 58 21 2D C4 CD : C3
C238 74 C2 CD 41 C2 0D 20 DC : 0F
C240 C9 C5 2B CD 6E C2 23 23 : FC
C248 CD 6E C2 3A 20 C4 06 00 : 21
C250 4F B7 ED 42 CD 6E C2 2B : 5D
C258 CD 6E C2 2B CD 6E C2 09 : 2E
C260 09 CD 6E C2 23 CD 6E C2 : 26
C268 23 CD 6E C2 C1 C9 7E BB : E3
C270 C8 3C 77 C9 CD 7E C2 73 : C4
C278 C9 CD 7E C2 5E C9 F5 D5 : C7
---------------------------------
SUM: 9A 5C 72 CF 7F 66 74 C9 FA4F

C280 08 3A 1B C4 5F 16 00 08 : 9E
C288 14 93 30 FC 83 F5 3A 20 : A5
C290 C4 5F 7A 16 00 19 3D 20 : 29
C298 FC F1 5F 19 23 D1 F1 C9 : 13
C2A0 C5 D5 E5 2A 18 C4 ED 5F : D1
C2A8 AC 67 AD 6F 54 5D CD C9 : 76
C2B0 C2 CB 3C CB 1D CB 3C CB : 83
C2B8 1D CB 3C CB 1D CB 3C CB : DE
C2C0 1D 7D 32 18 C4 E1 D1 C1 : 1B
C2C8 C9 3E 10 44 4D 21 00 00 : C9
C2D0 29 EB 29 EB 30 01 09 3D : 9F
C2D8 20 F6 C9 AF 32 FD C3 32 : B2
C2E0 FE C3 3A 1A C4 47 3C 3C : 98
C2E8 32 1F C4 3A 1B C4 5F 3C : C9
C2F0 3C 32 20 C4 AF 83 10 FD : 91
C2F8 32 21 C4 21 1C C4 5E 93 : 09
---------------------------------
SUM: F9 C0 44 4D C8 FE 40 07 2D51

C300 32 22 C4 3A 1F C4 47 05 : 81
C308 3A 20 C4 5F 16 00 21 00 : B4
C310 00 19 10 FD 22 23 C4 2A : 59
C318 1D C4 24 2C 22 25 C4 25 : 61
C320 25 25 2D 3A 1B C4 CB 3F : 9A
C328 85 6F 2C 2C 22 29 C4 3D : 98
C330 3D 6F 22 27 C4 2A 1D C4 : C4
C338 ED 5B 1A C4 7C 83 67 7D : 09
C340 82 6F 22 2B C4 3E 30 21 : 91
C348 2D C4 11 2E C4 01 43 01 : 39
C350 77 ED B0 3A 0F C4 13 23 : 57
C358 01 43 01 77 ED B0 2A 27 : AA
C360 C4 CD 1E 20 CD E2 1F 50 : ED
C368 20 3D 20 30 30 00 C9 21 : C7
C370 2D C4 CD 78 C3 21 71 C5 : 50
C378 E5 3A 14 C4 77 3A 1B C4 : 87
---------------------------------
SUM: 7A E8 54 A9 B1 96 27 77 A4E7

C380 3C 16 00 5F 3A 15 C4 19 : DD
C388 77 E1 E5 ED 4B 23 C4 3A : 96
C390 16 C4 09 77 3A 17 C4 19 : 88
C398 77 3A 13 C4 E1 E5 23 11 : 82
C3A0 01 00 CD D0 C3 E1 E5 ED : 14
C3A8 4B 23 C4 03 09 CD D0 C3 : 9E
C3B0 3A 12 C4 E1 E5 08 3A 20 : 38
C3B8 C4 4F 06 00 08 09 50 59 : D3
C3C0 CD C8 C3 E1 CB 21 0D 09 : 3B
C3C8 08 3A 1A C4 47 08 18 06 : 8D
C3D0 08 3A 1B C4 47 08 77 19 : 00
C3D8 10 FC C9 CD 18 20 E5 3A : F9
C3E0 1F C4 4F 2A 1D C4 CD 1E : 28
C3E8 20 3A 20 C4 47 1A CD F4 : 60
C3F0 1F 13 10 F9 24 0D 20 EE : 7A
C3F8 E1 CD 1E 20 C9 00 00 44 : F9
---------------------------------
SUM: B6 8F BA 78 1B 2F E9 4C E29B

C400 4F 4E 45 20 21 00 46 41 : AA
C408 4C 45 20 21 00 50 3F 23 : 84
C410 2A 58 49 2D 4F 4F 4F 4F : 34
C418 64 00 08 08 0A 0F 05 00 : 92
C420 00 00 00 00 00 00 00 00 : 00
C428 00 00 00 00 00 00 00 00 : 00
C430 00 00 00 00 00 00 00 00 : 00
C438 00 00 00 00 00 00 00 00 : 00
C440 00 00 00 00 00 00 00 00 : 00
C448 00 00 00 00 00 00 00 00 : 00
C450 00 00 00 00 00 00 00 00 : 00
C458 00 00 00 00 00 00 00 00 : 00
C460 00 00 00 00 00 00 00 00 : 00
C468 00 00 00 00 00 00 00 00 : 00
C470 00 00 00 00 00 00 00 00 : 00
C478 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: 29 EB B6 76 7A AE D9 B3 01E9
```

C480H〜C6B4Hまで0で埋める

```
C6B5 C9 C9                   : 92
---------------------------------
SUM: C9 C9 00 00 00 00 00 00 C9C9
```

▶現在，NHK総合で「ヤダモン」というアニメを5時50分から10分間やっています。短いものですけど僕は好きです。個人的には「ふしぎの海のナディア」よりも好き。人類史上最高傑作じゃないでしょうか（ちょっといいすぎ）。　清水　弘和(16)広島県

版下作成支援ツールＹ300-Ａ

版下作成支援ツールという特殊なソフトをレイアウトツールとして使ってみました。今回は発売版を使って処理の手順などを見ていきましょう。例によって今回の記事もＹ300-Ａで出力されたものを使用しています。

Nakano Shuichi　中野　修一

そもそも版下とは？

「版下作成」というのは特異な分野です。版下というのは印刷にそのまま使用できる原稿のことを意味します。

通常のページなら，私のようなライターの書いた原稿を編集者が整理し，レイアウターさんがページ内に綺麗に配置し，文字原稿は写植屋さんで活字にされ，図版はトレース屋さんが綺麗に仕上げて１ページ分の版下が作られ，それが印刷屋さんで製版され印刷される，という工程をたどるわけですが，このページに限ってはレイアウター以降の工程が印刷屋まですっぱり省略されています。これを世の中ではDTPというようです。

ソフトバンクの雑誌でもDTPが行われているものがいくつかありますが，これはさらに進んでいます。ライノトロニクスなどのイメージセッタで出力されたフィルムはそのまま印刷機にかけられるので，編集から印刷までの工程のうち製版の部分さえも短縮できることになります。時間と人手を大幅に削減できるのでDTPはこれからの出版界でますます多用されることになるでしょう。ただし，そうなると，それまで分業で行われていたものを全部しょい込まされる人も出てくるのですが……。

DTPツールとしてのＹ300-Ａ

Oh!Xの読者のなかに「版下」というものを扱う人がどれくらいいるか定かではありませんが，このツールは単に図版と文書を組み合わせて綺麗な書類を作りたい，という用途にも使用できます。数式や楽譜などを除けば（不可能ではないが……），たいていの用途で使用できます。もともと版下というのはもっとも高度な出力が要求されているものなのです（処理時間などを考えると必ずしも手軽にというわけではありませんが）。

現在X68000でこういった用途にあたる場合はMultiword，TeX，PressConductorといった処理系が使用できます。いずれも扱いやすさ，図版出力の制限，表現の自由度などの点で一長一短といったところでしょう。いちばん使いものになりそうなのがTeXというのも情けない話です。

処理の手順

それでは，Ｙ300-Ａを使った作業の手順を順を追って紹介してみましょう。

まず，版下の新規作成メニューから，作成する版下の大きさを指定します。プロッタを使用した場合2000×2000mm程度までの指定ができるようです。ごく普通のB5，A4，B4，A3程度であればプリンタでも出力できます。たとえば，Oh!XのようなA4変形判の場合でもユーザー指定ができるのでまったく問題はありません。

文書はあらかじめ作成しておき，大雑把なレイアウトも考えておくとよいでしょう。すべてＹ300-Ａ上で処理することもできるのですが，効率はよくありません。出力専用と割り切ったほうがいいでしょう。

まず，文字の部分の書体や段組を決定し，ファイルからテキストを読み込みます。流し込んだらイメージ通りになっているかひととおりチェックし，細かい部分の修整に入ります。

たとえば，見出し部分の書体を変える場合，書体設定部分で変更したい内容を登録し，マウスで範囲を指定して属性変更のメニューを選びます。属性変更は各項目ごとに独立に動作します。これは書体設定部分で設定変更を行っても，その直後に記入したものからしか有効でないことを示しています。ただし，書体指定の変更でただちに有効になるものがあります。文字データの張り付け位置，段組，字詰めなどです。

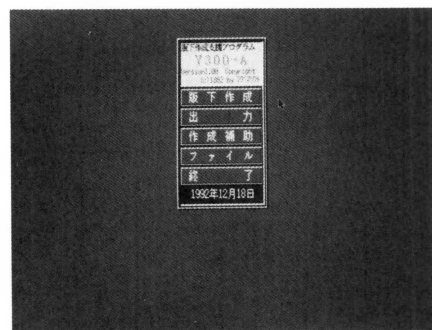
位置変更などは当然のことなのですが，字間と行間がひとつの文字データ群に対してひとつしか指定できないというのは多少注意が必要です。

見出し部分も文字間が変更できないので，大き目の書体に長体をかけて文字間をあわせます。もちろん，この文字データとは独立した指定をしてもいいのですが，行の移動などがあった場合に追従して動いてくれないのでここでは同じ文字データ内で処理しました。

文字データができあがったら見出し文字などを指定し，必要があれば図版を作成します。図版は部品として登録することで汎用に使えるので，あらかじめ作成しておいたほうがよいかもしれません。

あとはプリンタで打ち出して終わりです。印刷時に倍率を指定することもできるので，打ち出されたものを縮小コピーすることでさらに高い解像度の版下を作成することもできます。

前回の記事との違い

前回の紹介記事ではサンプル版を元に記事が書かれていたようですが，市販版では文字周りでの大幅な仕様変更がなされています。製品版ではある程度文字の大きさが小さいと文字の縮小表示を行わず，枠だけの表示になりました。しかし，拡大モードでのエディットが繁雑なため一概に改善とはいえません。もともと表示モードで枠だ

起動時の画面と基本メニュー

けの表示と文字表示は選択できるのですから，なにも強制的に枠表示にすることはなかったのではないかという気がします。

実をいうと前回のバージョンは文字エリアをひとつしか持てず，文字数制限も厳しく，さらに禁則処理も行われないという仕様でしたので，そう思って見ると前回の記事の出力例はかなり究極の使い方がされていることがわかります。

今回の市販版ではそのあたりが見違えるほど改善されています。まず文字エリアは8つ持つことができます。もちろん，それぞれで違う文字指定ができます。普通の文書を作る際にはこれで十分でしょう。禁則処理もサポートされました。追い込み禁則をした場合には，ちゃんと字詰めを変更し，均等配置にしています。

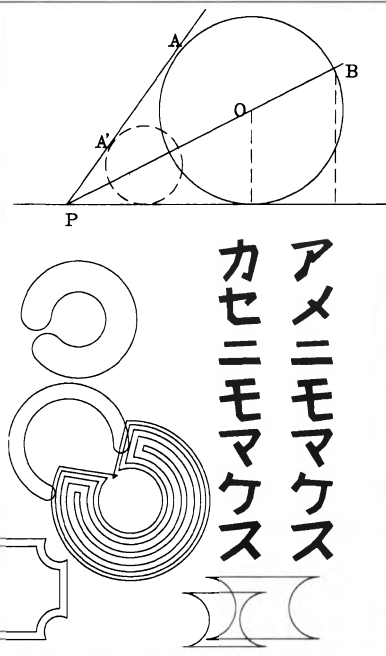
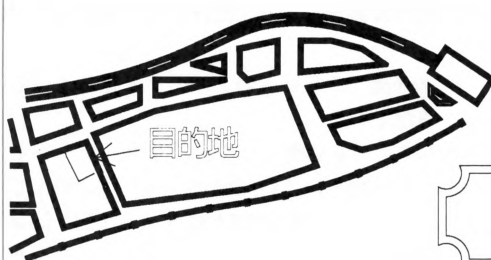
あとは図版の回り込みと字切り（要するに図版が割り込んできたとき，自動的に文字がよけるやつね）に対応しているとほぼ完璧なのですが……。

フォント指定の技

無指定だと全角文字と半角文字のバランスがいまひとつ悪いように思われます。横幅を単純に半分にしていることが原因でしょう。文字を等幅で半角をその1/2で処理していることは，それなりによいことではあります。変に気をきかせてくれるDTPソフトでは思わぬところでえらく面倒な作業を強いられることがありますので。

英数字の場合，通常の写植文字などでは漢字の7，8割の幅となっていることが多いようです。字詰めも詰め気味に出力されていることがわかるでしょう。日本語は等幅に英字はプロポーショナルにというのが一般的にいって美しい出力の姿です。

傾向がわかれば対策はそう難しくありません。たとえば前回の紹介記事では半角文字は本文の12級相当の大きさに対して16級相当の大きさが指定されています。ただしその場合は天地がはみ出してしまうので

書体指定ウィンドウ

図版などの出力例

平体2相当の処理を加えているようです。むろんすべて手作業で変換しています。

そのまま出力すると，「X68000用版下作成支援ツールY300-A」，「A quick brown fox jumps over the lazy dog.」のようなプロポーションになってしまうので，「X68000用版下作成支援ツールY300-A」，「A quick brown fox jumps over the lazy dog.」のように補正してやっているわけです。

Y300-Aではフォント指定の際に全角文字と半角文字を独立に指定できますので，半角文字についてはZSFNT.X（1992年6月号付録ディスク）などであらかじめ拡大しておいてやるという手もあります。

なお，書体倶楽部相当のものならなんでも使えるので，平木敬太郎氏の作成したベクトルフォントを御木徳高氏のコンバータで変換したものを使用してみました。

すでにこれらのフォントは提供されているのですから，こんなふうに混在させた文書を作りたいと思うのはごく当たり前のことでしょう。なんでこんなことがいままでちゃんとできなかったんですかねぇ。

印刷設定を行う

最後に

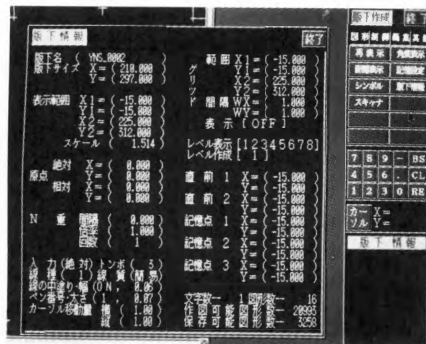
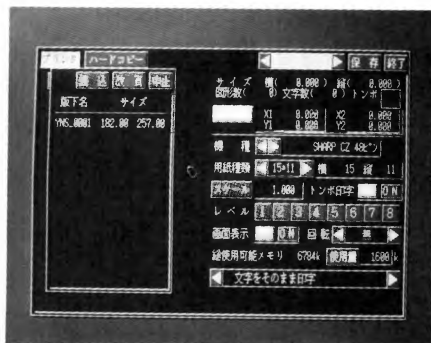
図版ではもっと面白そうなこともできるのですが，絵心のせいでいまひとつ能力を発揮できなかったようです。

Y300-Aの図版はすべてドローイングデータとして扱われます。周辺機器としてハンディスキャナがサポートされていますが，基本的にトレース用の画像読み込みのためのものです。高解像度の出力ではビットマップ画像もそこそこの画質で出力できるはずなので，モノクロ画像のビットマップ出力くらいはサポートしてほしかったところです。

なお，Y300-Aは当面，通信販売のみとなります。普通のソフトのように店頭デモなどを頼りにすることができないのでなかなか判断しにくいことと思います。全体的な完成度はまずまず。ユーザーインタフェイスを含め，改善してほしい点もいくつかありますが操作性は必ずしも悪くありません。問題は処理速度です。

フォントつきでプリンタさえあれば，即，使えます。とりあえず手間を惜しまねばちゃんとした出力が得られます。

現在主流となっているCZ-8PC5やBJ-10などの48ドットプリンタは360dpiの解像度を持っています。これはレーザープリンタと同等の解像度です。48ドットプリンタの普及率が異様に高いのでこんなプログラムももっと普及してほしいのですが……。

X68000用　5'2HD 2枚組　29,800円
マグマソフト　0992(68)2286

X68000 次世代へのかけ橋

ユーザーの期待とシャープがなすべきこと

Saitou Susumu 斎藤　晋

発売以来6年にわたってユーザーとともにパーソナルコンピュータのあるべき姿を追ってきたX68000。モデルチェンジへの期待が高まるなか，世界の動向を見据えつつ要望を整理してみよう。

今年こそは，と待っている人も多いだろう。もちろんX68000の32ビット機のこと。初代機が登場して以来6年，X68000のハードウェアはXVIでクロックが10MHzから16MHzになったこと以外には大きな変更はない。ユーザーの気持ちとしてはさすがにもう待ち切れないというのが正直なところだろう。いったい新しいX68000はどのようなかたちで姿を現すのだろう。ここで簡単な予想を試みてもよいのだが，当たっても外れても，表面的な部分に目がいくだけで，かえって本質を見逃しかねない。ここでは，これからのX68000のありようを考えるために，いろいろと周囲の状況を見渡しながら，私たちがシャープに対してなにを要望すべきかを書き連ねてみたい。

みんなの要望？

さて，まわりのX68000ユーザーに新製品に対する要望を聞いてみると，だいたい同じような答えが返ってくる。その1つひとつはもっともな要望だが，それらすべてに応えていると，非現実的な内容になってしまうかもしれない。

たとえば，CPUに68030の25MHzとか，コプロセッサの標準装備だとか，メモリは4Mバイト以上とか，SIMMのソケットは8つは用意してねとか，ハードディスクは高いから外付けでいいとか，フロッピーディスクは5インチと3.5インチを両方ともねとか，3.5インチは3モードがうれしいとか，CD-ROMもあったらいいなとか，MOは安くならないのとか，それから拡張スロットは4つ付けてねとか。ふう。

よく出るのはグラフィック表示に関する要望だ。ノンインタレースで1280×960ドット程度のハイレゾにしてとか，色数は1670万色のフルカラーでしょとか，そうなるとやっぱりディスプレイは17～21の大型でないとだめだとか。それから描画を高速にするアクセラレータがいるとか，グラフィック画面でのドットのアスペクト比を1にしろだとか。

そしてAV機能もてんこもり。サウンドはやっぱりPCM音源かなとか，DSP積んでねとか，いっそMIDIも標準にしてSC-55を内蔵したらとか，ビジュアルといったらビデオの入出力もお願いねとか，SX-WINDOWでテレビが見たいなとか。トースターも欲しいなとか(?)。

とまあ，いろんな人が勝手なことをいうわけだが，私の意見としては標準スペックよりも拡張性を第一に考えてもらいたい。たとえば，メモリ。メモリはSIMMでないと困る。はっきりいってメモリのような標準部品はわざわざパソコンメーカーから買うものではない。たとえば，現状でX68000の増設メモリはXVIの純正品だと4Mバイトあたり114,600円もする。コンパチ品でも7万円くらいだ。これが，そのへんで売っているSIMMなら4Mバイトあたり2万円くらいで買えるのだ。また，グラフィックVRAMも2Mバイトまでは増設できるようにしてほしい。

というわけで，ハードウェアに関してはもうとっくに完成しているだろうから，いまさら要望を書いてもしようがないだろうけどね。とにかく出てきたものをどう扱うかってとこでソフトウェアの問題が重要となるだろう。

私が聞いたなかでいちばん困った要望はMacintoshとの互換性をというものだ。はっきりいってMacintoshの環境がほしいならMacintoshを買えばいい。というのが私の意見だ。確かにMacintoshには見習うべき点も多いが，純粋に真似をする必要のある部分はごく一部にすぎない。Macintoshでしかできないことの多くは，Macintosh本体ではなく周辺機器によるものが大半だからだ。

それよりは，データ互換をしっかりやってもらいたい。Macintoshから持ってきたいデータというのは主にグラフィックやサウンドデータだろう。それはMacintoshに限らず，どの機種からの場合でも同じく重要なことだろう。

マルチメディアと動画

X68000はテレビ事業を母体に出てきたこともあって，当初からマルチメディアを意識したパソコンだったといえる。しかし設計者の思想を私なりに察すれば，マルチメディアなどというキーワードを意識したわけはなく，もともと絵も音も私たちのまわりにあるごく自然なデータとして扱えるべきだという感覚なのではないだろうか。当然のことながら，絵や音のデータを自在に扱うには，CPUパワーだけでなく，さまざまなコンピュータ資源が必要となる。特に動画や音声をデジタル信号として処理するためには，大容量のメモリや，外部メディアが必須である。また，動画などでは圧縮伸長などのデータ処理が要求され，ハードウェアの助けも借りなくてはならない。

X68000が登場したころには，CD-ROMもまだまだ一般用途に利用できる段階ではなかったし，初代機ではハードディスクさえ内蔵できなかった。フロッピーディスクで起動するのが当たり前の時代である。

マルチメディア時代のX68000の標準的な環境としては，メモリ16Mバイト，ハードディスク1Gバイトは当たり前となるだろう。3.5インチのMOディスクも数枚持ち歩くことになるのではないだろうか。

マルチメディアというと，やはり動画はどうするのかといった話にならざるをえない。X68000ではDōGA CGAの活動によるアニメーション作品の制作が盛んに行われている。これとは別にシステムレベルで動画を扱いたいという要望も大きい。

この分野でもMacintoshがQuickTimeの発表で一歩リードした感じだが，Windowsでも新たにVideo for Windowsが発表され，話題を集めている。

QuickTimeとVideo for Windowsの違いを見るとMacintoshとIBM系マシンの環境の違いを反映していて面白い。

まずQuickTimeのムービーファイルは，使用する機種のモードに応じて忠実に再生できる。Macintoshの場合，使用する機種や状況によって表示色がモノクロからフルカラーまでまちまちだが，すべての機種やモードでデータを共有できるため，データを作る際に再生する状況を考える必要はない。しかし，実際にはモノクロの機種でカラーのムービーを再生する場合にはディザによって色の違いを表現しようとするため，CPUにかかる負担は大きく，なめらかな動画は難しい。逆にデータがモノクロであってもフルカラー表示の場合には表示は楽にはならない。

一方，Video for Windowsの場合は，かなり現実主義に根ざしている。もともとIBM系マシンではMPCの規格として256色以上であることを条件としているため，16色モードでの表示のことは考慮されてない。動画を見たいような人は256色モードで再生しなさいということだ。また，ほとんどのグラフィックカードの表示能力は256色までだから，動画データもそれを基準に作成されることになるだろう。また，動画処理にはインテルの圧縮伸長技術が利用されているが，これをハードウェアで行う専用のアクセラレータも準備されている。

さて，動画を扱うプラットフォームとしてはX68000は申し分ないハードウェアであるといえる。となると，問題は規格をどうするかというところに最大のポイントがある。QuickTimeとVideo for Windowsの例を参考に十分な検討を重ね，X68000に最適な方法を採用してもらいたいものだ。いずれにしても重要なことは，SX-WINDOW上のアプリケーションから動画を扱うための規格の策定とそれを実現するソフトウェア，そしてビデオ画像のキャプチャリングや圧縮伸長のハードウェアなど，周辺装置のサポートということになるだろう。

ネックは周辺機器

X68000の今後を考えるうえで，メーカーであるシャープに要望したいのは周辺機器のサポートである。以前から指摘されてきたことだが，はっきりいってX68000最大の弱点といってもいいだろう。

Windowsで羨ましい点は周辺機器が多種多様にあることだ。たとえば，プリンタにしてもたいていのプリンタメーカーはWindows用のプリンタドライバを用意している。そのためWindowsアプリケーションは自力で各種プリンタをサポートする必要がない。SX-WINDOWでもPC-PR＊＊＊やESC/Pなどのプリンタをサポートしているが，ポストスクリプトやレーザーショットのPDLはサポートされていない。Macintoshのソフトなどが移植されてもページプリンタがなければ魅力は半減してしまう。プリンタメーカーが自主的にX68000をサポートしてくれるとは思えないので，ここはシャープが頑張るしかないだろう。

本当はネットワークについても触れたいのだが，イーサネットカードもIBMマシンだと２万円弱で買えるのに，X68000の場合はちょっとね。基本的にはOSの問題なのでこれがまた奥が深い。

SX-WINDOWの環境を

Windowsなどを使ってみると，そのユーザーインタフェィスには山ほど疑問がわいてくる。その点SX-WINDOWのGUIは実にわかりやすく，かつ作業効率がよい。

しかし，使いやすさはともかくSX-WINDOWにはまだまだアプリケーションが少ないのが辛いところだ。それから，かな漢字変換は早くインラインで変換できるようになってもらわないと困る。ASK68Kはバージョンアップも必要だが，その前にシャープからSX-WINDOW上でのかな漢字変換の標準規格（アプリケーションからのファンクションコールやワークエリアの使い方について）を策定し，それに合わせてASK68Kのニューバージョンを作ってもらいたい。そうすれば，アプリケーションはASK以外のかな漢が出てきても意識することなく使用できるようになる。

またSX-WINDOWではアプリケーション間の有機的なオブジェクトリンクの方法を確立してもらいたい。ワープロの文書に音楽データを貼り込んでおけばマウスのクリックでプレイヤーが起動できたり，電子メールにアニメーションデータとPCMによるメッセージを貼り込んで送ることができたり……。こういったことをアプリケーションを問わずできるのがウィンドウ環境の理想ではないだろうか。

世界のPCはいま？

この１年ばかりのパーソナルコンピュータの状況を見ると，最も目につくのはDOS/Vマシンの台頭だろう。ご存じのように，DOS/VはIBM系のパソコンでそのまま日本語を扱えるようにしたMS-DOSのことだ。以前にも，IBM互換機をベースとしたAXシリーズなどが話題となったが，あれはハードウェアを付加することで日本語対応とする日本独自のシステムであった。DOS/Vでは漢字ROMすら使用せず，すべてソフトウェアで対応するため，海外の安い機種がそのまま使用できる点で大きく異なる。結果として安くてハイエンドなマシンがどんどん国内に入ってきたわけだ。

確かに，DOS/Vマシンの価格と性能には目を引くものがあるが，表面的なスペックと価格でパソコンの価値を判断するのは危険な傾向だ。ましてユーザー指導型の市場になってきたなどと書きまくる一般のマスコミの誤解には困ったものである。

IBMマシンが極端に安くなっている理由は以前にも桒野氏が書いていたが，要するに互換部品を専門に開発するメーカーの厳しい競争によって，開発力をもたない弱小メーカーでもIBM互換機をつくれる（組み立てられる）ようになったことが大きい。大手メーカーどうしの競争なら価格を崩壊させるような首の絞め合いはなかったろう。そして，ついには本家のIBMまでが，コストの高い純正部品を使わず，世界標準部品（互換部品のことをこう呼ぶらしい）を使いはじめたというわけだ。もはやIBM自身がAT互換機を販売しているといっても過言ではないのである。

もちろん標準化によって価格が下がることはいいことに違いない。しかしこの動きは技術の進歩と逆行している面がある。まず，AT互換機というのはIBM ATと互換性をもつマシンで，ISAバスのマシンと理解しておいてほしい。ATは仕様がオープンにされていたため，ATバスはISA（Industrial Standard Architecture）として互換機の間で標準規格となった。そこでIBMはPS/2以降，ISAよりも高性能なMCAという32ビットバスを採用し，この仕様をクローズにしてしまった。対する互換機メーカーも共同でEISAバスを開発し，これをAT互換機の次世代製品にしようとしたのである。結果はどうなったか。

MCAマシンもEISAマシンも技術的には注目されながら，どちらも主流には至っていない。むしろ値下がりを続けるAT互換機に人気が集中しているのだ。安いだけでなく，ATには膨大な周辺機器(とりわけグラフィックカードが重要)がつながることもある。結局IBMも古いATを押し入れから引っ張り出すことになってしまったわけである。

ATのバス幅は16ビットで動作クロックは８〜10MHz，とても486などの32ビット/33MHzといったCPUの能力をまともに生かすことはできない。最近は，グラフィックのパフォーマンスを上げるためにローカルバスと専用のグラフィックアクセラレータを搭載したマシンが出てきたこともあり，いかにATバスが足を引っ張っているかが話題となるようになってきた。いずれにせよ，世界のPCも古いアーキテクチャに縛られているのである。

ANOTHER CG WORLD
ムクムク
ムクムク
Feb
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28

むぎゅ

小さくたたんで…

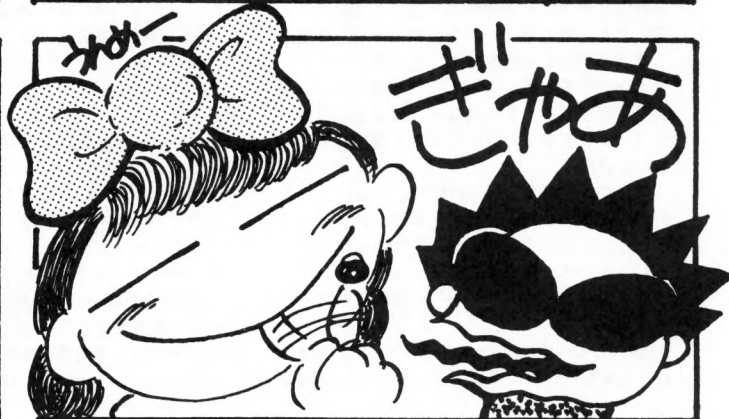
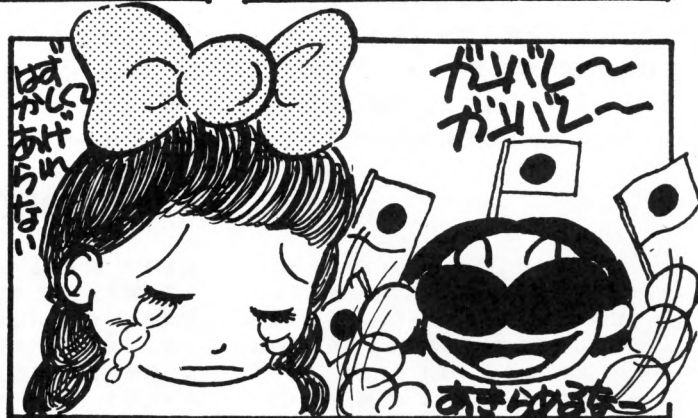
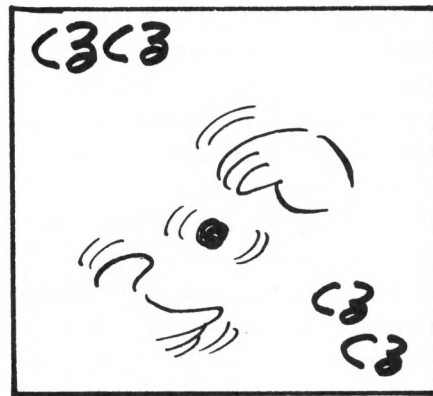
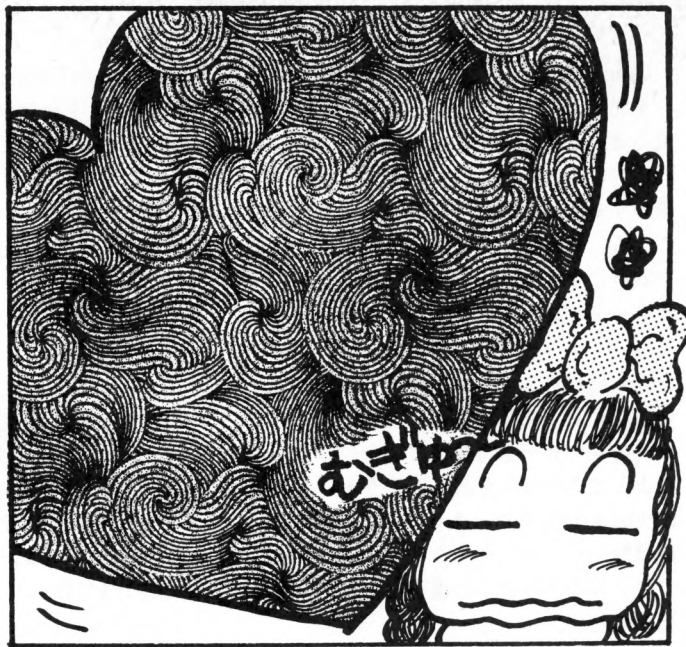
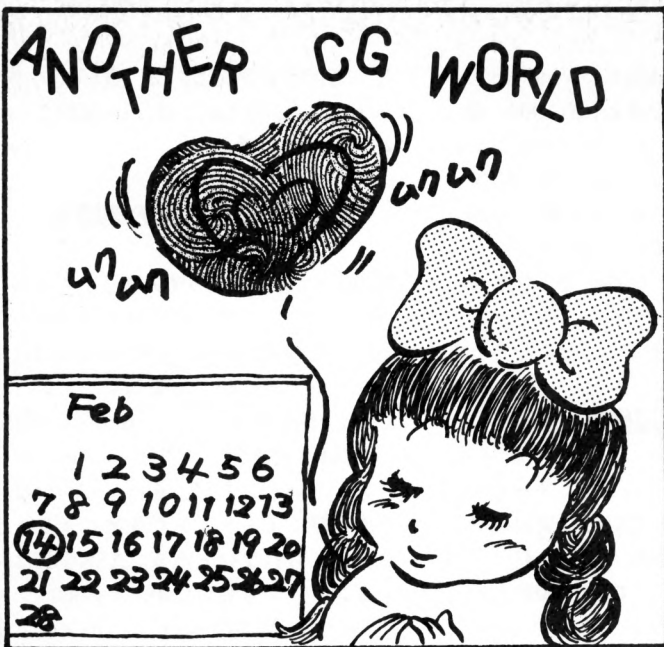
あこがれのあのひとにプレゼント…

くるくる
くるくる

こんなんなっちゃったー

はずかしくてあげられない
ガンバレ〜ガンバレ〜
あきらめるなー

マジマジ
ドキドキドキ…

ぎゃあ
元も子もないよねー これじゃー

どどーん！
CG画像
RGB
ベタファイル
うんせ〜
重たいよー
小さく
なーれ
PICにして保存
X68000上で見る
ならこれで充分
CGは画面で見る
のがやっぱり
キレイ
ポジにして保存
マシンがなくてもいつでも
どこでも見られる
写真ハガキなんかにも
応用できる
でも、いちばん
大切なのは
心に映像が焼きついて
いるってことよねー
今回のCGデータだよん
総物体数 554 うち光源2
メタボール数8
1280×1024 ピクセルをフルカラー
で4×5ポジに出力
使用ソフトは C-TRACE
COPYRIGHTの文字は
サイクロイドで作成して
合成

「長島が監督になる年は，いつも最不況なんですよ」

京橋にあるD証券東京支店の，広いロビーの一隅を仕切った応接室で，次長の永井さんはコーヒーをすすめながらいった。

「エスプレッソですよ，私たちもお客さまのときしかいただけないんです」

ガラス越しのビル街は，暖かな12月だ。

見てないお金

「前回，長島が巨人の監督に就任したときも，景気がどん底の年だったんです。チームも景気も上向きになるころ，やめることになりましたけどね。今回もまた長島が出てきましたから，これからは景気は上向きですよ」

無条件にまわりを明るくする長島さんがもとめられているときは，よっぽど不景気のとき。そして，どん底までいった景気は上に行くしかないということか。

大黒さまもビックリするような「福耳」の永井さんが，エビス顔で景気回復を予言するさまは，なかなかCM風である。

「景気なんて，不況だ不況だとみんなでいうからますます悪くなる，そういう影響もバカになりませんよ」

「景気」は，まず「ありさま」「光景」のことだと国語辞典にもあった。だとすると朝晩のニュースで「最底値」だ「不況」だと伝えているありさまが，いちばん「不景気」なのではないか。

D証券には，わずかばかりの額だが長いこと任せきりにしてあるお金があって，いつのまにか6つくらいの銘柄に姿を変えている。もともとよくわかってはじめたわけではなく，銀行の定期預金と同じような感覚であずけていたものだ。その間，すすめられるままに「売却」と「買い付け」をくりかえして，もう7，8年になる。それが今回の不況のあおりを1人前にこうむることになったのだ。

運用の推移を報告する伝票はひんぱんに送られてくるけれど，お金はあずけたとき以来見ていないので，損も得も実感がうすい。10,000円が6,000円になったと書いてあっても，目の前には1円もないのだからゼロになったとしても同じなのだ。

誰かに貸した本が友人のあいだを又貸しされて渡りあるき，だんだんボロボロになって，ことによるともう返ってこないかもしれない。でもあんまり長いこと手もとにないので，それでもしかたがないような気になってくる。そんな感じだ。

「世間がみんなそうなんだから，しかたがないよね」

なんて夫と話しているところへ，D証券から，またあらたな銘柄の入手をすすめる電話があった。

気がつけば野球

このところ「売り」「買い」のすすめがいっそう多くなり，そのつど説明もあるのだが，ほんとのところ誰のために好都合なのかよくわからない。「売り買いして手数料をかせぐのが証券会社だからね」と夫がいうのを聞くと，いいなりになっているのはオメデタすぎるとも思う。

そこで，いつも手続きのときは投資係の女性，松岡さんが自宅まできてくれるのだが，今回は私が出向くことにした。

「すこし，この分野も勉強してみたら」と夫にいわれて，たまには店頭のようすものぞいてみようかと思ったのだ。それにいくらかでも積極的な姿勢をしめすほうが，自分たちにも有利かもしれない。

ところで，むかえる側もぬかりがない。とくに例の損失補てん事件のあと，お詫びと心機一転の旨を新聞に広告してから，前にもまして丁重だ。

たかが顧客のひとりが来店しただけなのに，コーヒーでもてなし，次長さんが顔を見せてリップサービスにつとめる。

話のほうも，長島監督が登場するまでは，このところの相場の問題点や，地価や資産の算出のカラクリなどで（任天堂商法なんて言葉も出てきて），報道で聞く内容とはひとあじちがうものだ。

じつは私は耳をかたむけているのが精いっぱい。同席していた松岡さんも，ときおりいたわるような視線を送ってきた。

長島さんの話になったとたん，われながらホッとしたのがわかった。エモン掛けのようだった肩のツッパリがとけて，口もともリラックスしてくる。やっぱり無理は身の毒，自分にわかる話がいちばんいい。

でも，意外にも私よりゴキゲンになったのは永井さんのほうだった。

「長島が現役引退後，はじめて監督としてカムバックしたときは，うれしくてほんとに涙が出ました。私は狂がつくほどの長島ファンですから」

こっちもおなじ。「野球は巨人」の時代は，すくなくとも子供はみんな巨人ファンだった。巨人ファンをいいかえれば，長島と王のファンだった。夜になると後楽園球場の照明が空をそこだけ明るくし，新宿の家からもよく見えたものだ。

「とても愛すべき純粋なかただそうで，ずいぶん愉快なエピソードや伝説もたくさん聞きますね」と私。

「単純なところがありますからね。でも彼は若い者を育てるのがじょうずですよ。ただ運があまりよくない。おいしいところをみんな誰かに取られるんですよ」

「そうですね，けっきょく前のときも藤田さんにゆずってしまいましたものね」

「そうそう，こんども誰かに取られると思いますよ。あるいは王かもしれないし」

仮にもお金を託しているのだし，きょうはむやみに笑顔を見せまいと心をひきしめて出かけたのに，やっぱり談笑で終わってしまった。けっして受けとるまいときめていたオミヤゲも持たされ，けっきょく「よろしくおねがいします」なんていい残して

帰ることになったのだ。
礼儀正しい指図にしたがっていくうちに気づいたら身ぐるみはがされて野原に立っていた，宮沢賢治の「注文の多い料理店」をなんとなく思い出しながら。
つぎは，トリックを知ってしまったものの，笑いとばすわけにはいかない話。

予備校データベース

不況のせいで，悪徳商法が増加しているという新聞記事があった。
まったく，あの手この手で何かをもくろむ人たちから，連日さまざまな電話がかかってくる。
「無料の試写会にあなたがモニターとして選ばれた。きてくれたら3千円さしあげる」とか，「無料でグルメの試食会があり，参加すると記念品が出る」とか。
「無料」の埋め合わせは，かならずどこかに伏せられているはずなのだが。
悪徳というのではないが，今回，ちょっとしたセールスのウラ側を，かいま見ることになった。
もうこれまでに何回も電話をかけてきては利用をすすめる「S」という会社があった。予備校の情報を受験生に提供し，志望大学，学部への合格のために，ふさわしい予備校，塾を紹介するというものだ。
「私たちは，予備校ではありません。予備校を紹介して，入学がきまったら予備校側から手数料をいただきますので，お客さまからは料金はいただきません」
予備校の情報を網羅しているかのような自己紹介であり，その提供にあたっては，公正中立の姿勢であるような印象だ。情報源は，じっさいに予備校に通っていた多数の現役大学生であるという。
そんなビジネスがほんとうに成り立つものなのか。短い間の知識だが，予備校は知名度のある多人数制のもの，できてあまりたたない少人数制のもの，その中間のものに分かれるくらいで，内容的には一長一短，大同小異だ。そんななかで責任をもって示せる情報とはどんなものなのか。
また，自力でじゅうぶん生徒を集められる学校にさらに生徒を紹介するとは思えないし，そういう学校が紹介料を支払うはずもない。となれば，生徒集めに苦慮しているところと特約していると考えるのがしぜんだ。
高2のトオルへの予備校からの勧誘はあいかわらずで，すこしとぎれたかと思うとつぎの波がくる。トオルはすでに志望大学をきめていて，いずれはそのためにどこかに通うつもりでいる。
じっさいの勧誘は，数のうえでは郵便によるもののほうがずっと多く，郵便で案内を送ってくる予備校が電話をかけてくることはない。電話をかけてくるところは，創設も新しく知名度もないところ，そして少人数制の塾が多い。
知名度の高い予備校は生徒がよく集まり，そのために1人あたりの授業料も比較的安い。システムが完成されていて，模擬試験なども定期的におこなわれ，統計上の資料も豊富にあたえてもらえる。ただし，そのぶん生徒1人あたりが受ける学習上の利益は，逆に小さくなるだろうということは想像できる。
その点，知名度はゼロでも，個人的な要望と実力にあわせた指導をしてもらえるなら，少人数制もいい。ただし，授業料がだいぶ高く，実績もはっきり出ていない。
とくにこのところ，たいへん強力な勧誘をしてくるところが1，2ある。
そこで思いついたのが，例の「S」という情報屋さんだった。
どうせ，マイナーなところにあっせんする商売だと思うけれど，どんな話をするかいちおうきいてみよう。どの道，勉強するのは本人で，予備校はどこも大差なしだ。

専属セールス業者

「S」社に電話すると，担当者が自宅まで説明にくるという。
約束の日の午後7時，50歳前後の小柄な男性がおとずれた。明るく人あたりのよい感じで，しかも活発そうだ。そして椅子に腰をおろすやいなや話をはじめた。
話はすべて，S社専用の便せんに図をかきながらの熱弁で，なんと3時間。
内容は，もとより予備校を利用しないといかに不利かを説くものだから，すくなからずコジツケがある。わざわざ来訪して教えてもらうほどの話ではないのが実感で，とても忍耐がいった。
やっと予備校にはどんな種類と学習方式があるか，それぞれどんな特徴があるかの話までたどりつき，「のぞましい予備校」のスタイルが結論づけられる。それは少人数制で現役の大学生が指導にあたるところ「Tゼミ」だ。彼はカバンからTゼミのパンフレットを取り出して，くわしい説明をはじめた。

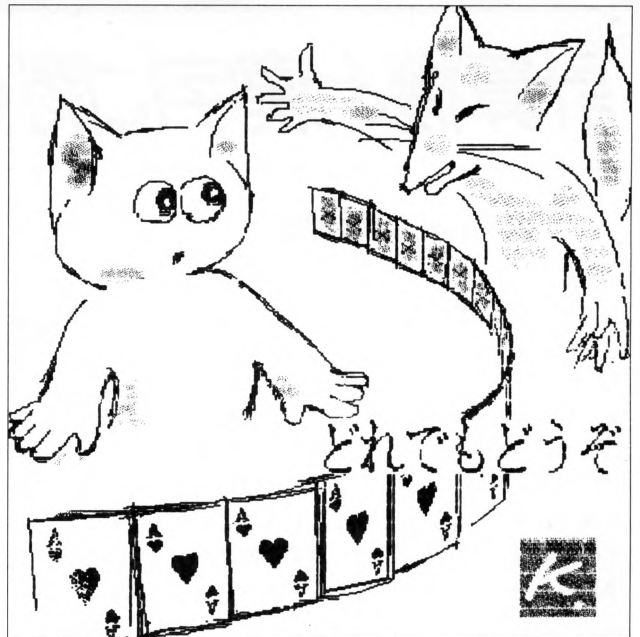



illustration Kyoko Takazawa

「Tゼミ」は新宿に6，7年前にできたそうだ。ぜひ親子で見学にいくとよいとすすめられ，訪問の予定日もだいたい指示された。相手に連絡をとっておくという。
話の運びかたが終わりに近づくにつれて不しぜんになった。トオルは，自分の直感から通いたくないときめた。2人で見学にいく必要もなくなった。
ことわりのためにS社に電話をしたら，当の担当者は不在だった。それではと直接Tゼミに電話をしたら，誤って「T本部」というところにかけてしまったらしい。
「失礼しました。Tゼミの電話を教えていただきたいのですが」
というと，こんどは相手がまちがえて，S社の電話番号を私に伝えた。
「それはS社ではありませんか？」と私。
すると，相手のいうことには，
「SはTゼミの営業ですから，話は通じると思いますよ」
なんと，やはりS社というのはTゼミ，あるいはほかのいくつかの特定の予備校専属の営業部門だったというわけだ。
情報を提供するというふれこみがあまりにおかしいと，トオルと2人でおおいに笑ったけれど，いずれどこかに通うとなれば，たいしたちがいはないだろう。
今はまだダマされたわけじゃないけれど，近いうちに，どこにダマされるかをきめなくてはならない。

計算機と漢字に関するタブー

漢字TALK7の登場

Macintosh用の新しいオペレーティングシステム「漢字TALK7」が出ました。もちろん，いままでも英語版システムを日本語化したものである「漢字TALK」は存在し，いちおう，並みの国産パソコン程度には日本語が使えたわけですが，今回の登場は，いままでのバージョンアップとは格段に違う意味合いを持っているということができると思います。いちばん重要な点は，多国語対応にするために根本からオペレーティングシステムを設計し直したということです。

従来は漢字TALKという名のついたパッチを英語版システムにあてることにより，なんとか日本語化していたといえます。したがって，漢字TALKとほかのさまざまなアプリケーション(特に起動時組み込み型ソフト)とのあいだで問題が起きること(パソコンが起動しなくなる，すぐダウンするなど)も珍しくありませんでした。また，漢字の表示や印刷についても統一した設計はなされていませんでした。

漢字TALK7では，これらの問題をかなりの程度まで解決したといってよいと思います。特に漢字の印刷と表示に関しては，「漢字TRUE TYPE」という新たなフォントベクトル表示方式を標準で採用することにより，5種を越える漢字書体を拡大も自由に画面やプリンタに出力できるようになりました。これは日本語デスクトップパブリッシングの新たな進展を確実にうながすことでしょう。

というように，日本語の障壁という問題を何年もかかってクリアしたはずの漢字TALK7ですが，誰もがさっと使うことができるかという問いかけに対しては，残念ながら「YES」と即答することはできません。漢字TALK7をインストールした場合，ハードディスクをなんと50Mバイトも使うのです。実に，そのうちの40Mバイト以上がフォントだというのですからたまげたものです。メモリも，8Mバイトぐらいは欲しいようですし，さらに，機能を拡大したために，前のバージョン(漢字TALK 6.0)に比べてスピードが20%程度遅くなってしまったので，68030以上のCPUを搭載したMacintoshが望ましいようなのです(そういうわけで今後のMacintoshのラインナップには68020以下のものは含まれないのです)。

漢字を攻撃する言語学者

「漢字TALK7によって我々のかかえる日本語あるいは漢字といった問題がどこまで解決されたのか？」

この問いを否定的な方向に根源まで突き詰めると，「日本語は計算機にとって本質的な障害である」ということになるのかもしれません。この問いに対して，明確にYESと言い切り，それどころか，漢字を計算機の世界から排除すべきだというところまで主張している外国人の言語学者アンガーがいます(参考文献)。

アンガーはまず，日本語という言語自体あるいは日本語文化というものから，日本語の表記方法という技術的な問題を分離することから説き始めます。日本文化を欧米文化に同化させるということではなく，単に技術的な問題であるという点を強調することを意図するからです。

彼は，もともと自然言語では話しことばが本質であり，表記とは，単に話しことばを反映しているだけだとします。字を読んだり書いたりできない人でも日本語そのものの理解に関して本質的な違いはない，現在表意文字といわれる漢字でさえも人間の脳における処理の過程では必ず音声情報に直されている，などの例を挙げます。

その次に彼は，日本語の現在の表記というものがいかに計算機にとって困難であり，計算機の処理に本質的に向いていないかということを数々の具体的な研究例を題材にして浮き彫りにします。そのような批判の基盤をなしている思想は，哲学者J.A.サールの定義した「強いAI」の否定です。要するに，知能はすべて記号情報の形式的操作におきかえられるという，楽観的であり，人工知能研究者が(とりあえず)のっかっている思想です。

アンガーは，言語自体や文化などと表記の問題を最初に分離したはずなのですが，このあたりにくると，漢字への攻撃は，計算機に関わることだけでなく，きわめて広い場面におけるものとなり，ますます激しさを増します。

いくつか例を挙げます。

・教育では知識の詰め込みばかり(国語の授業で漢字の読み書きの練習に時間をとっているので，作品の意味や文体，美しさを論じる時間がない)
・出版物の質的低下，マンガは漢字の困難さからの息抜き
・筋の通った論理についていけない
・有名人のいうことばかりうのみにする

もちろん，これらのことが漢字と直接関係があるとはしていないのですが，暗にその影響を示唆するのです。もしかしたら，この論調は保守的な日本人の感情を刺激して，せっかくの技術的な提案における説得力を逆に減らしてしまうという影響があるかもしれません。

漢字問題の解決方法

アンガーは，漢字の表記の困難さ，計算機との本質的な不適合を指摘するだけでなく，今後計算機における表記に関してとるべき道筋を具体的に示します。

1)　アプリケーションのうち，漢字かな交じり文の出力を絶対必要とするものと，そうでないものを分離する。
2)　次に前者の，少量のアプリケーションにともなう無連想式入力の国内規格を決める。
3)　後者の，多量のアプリケーションではローマ字を使ったデータ処理を行うようにすすめる。

これにより，日本において計算機の障害となっているすべての問題を解決することができるというのです。彼の主張はわかりやすく書くと次のとおりです。使わなくてもすむアプリケーションでは，なるべく漢字表記はやめましょう，どうしても使わなくてはならないワープロなどでは入力は無連想方式にして計算機の処理を軽くし，必要のないアプリケーションでは，日本語を使いたいのなら，ローマ字にしましょう。

「無連想式入力」の説明をしなくてはなりませんね。これは，各文字を一定の規則でコード化し，それを直接キーボードから指定して呼び出すというもので，引き出される文字の意味や読みなどと直接関係のないコード化をするところがミソです。いずれにせよ，変換式の入力法が主流となっている現在，これはかなり異端の主張といえるでしょう。

無連想式の入力方式の例として，東京大学の山田尚勇教授の提案による入力法が紹介されています。それは，漢字やひらがなを一文字一文字直接指定するものです。キーボードは左手と右手に分かれており，それぞれ五本指分の５つのキーが４段並んだものとなっています。そして，２回順番にキーを押すことにより１つの文字を指定します。こうすれば，組み合わせとして(４×５×２)×(４×５×２)＝1600通りですので，最大1600文字の指定ができるわけです。

知能機械実現への道

アンガーの提案に関しては，まず，無連想式入力は本当に現実的なのか，という疑問がわきます。最初に修得するまでは少し時間がかかっても，慣れたら漢字の入力は楽だということなのですが。この点については山田教授が書かれている論文などを読んで検討してみないと，直感で話してもしかたないでしょう。

一方，多くのアプリケーションでローマ字を使用しよう，という主張には大きな疑問を感じます。僕は漢字の交じらないひらがなで問題ないのではと思うのです。見やすさはローマ字に比べれば段違いによいと思います。ひらがなを加えるぐらいの拡張ならば，技術的に大きな問題はないでしょう。英語圏の人はローマ字のほうがなじみやすいでしょうが，アンガー自身がいっているように，言語自体と表記の問題は別であり，ローマ字にしたからといって，日本語が理解しやすいということにはあまりならないでしょう。

ただし，アンガー自身も必ずしもローマ字にこだわっているわけでもなく，音素で表せばよい，という主張ですので，ひらがなでということは，彼の主張と大きくへだたるということはないでしょう。

アンガーの議論は大筋ではここで紹介したような流れをたどりながらも，学術的なデータ，ほかの専門家などの意見，政治的な動向などを取り込みながら，重層的な構造になっています。本書ではまた，いまではほぼ収束しつつある国家的プロジェクトをになったICOT(新世代コンピュータ技術開発機構)の批判といった面も強調されています。漢字の取り扱いをICOTの動向と直接からめて議論した人も珍しいといえましょう。

コンピュータを過信して，すべて計算機に負担をかけていこうとするアプローチよりも，時代の流れや計算機の進展に合わせて，人間のほうもそれなりに変わっていくべきだという発想は，実は，強いAI思想に基づいて研究をしていく状況においても，迂回しているようにみえて実は最短距離となる場合もありうるのではないかと僕は本書を読んで考えさせられたのでした。とにかく面白い本ですので，ぜひ読んでみてください。

参考文献

マーシャル・アンガー，言語学者が見た第５世代コンピュータ「コンピュータ社会と漢字」，サイマル出版会，1992年９月

図１　山田式コード入力法(２打鍵コードによる)

漢字コード表

左側が漢字コード表の全体である。この表の１マスがそれぞれ５個×４段のマトリクスに対応しており、左側に示すように、それぞれのマスが文字を表している。したがって、このコード表では全部で1600（つまり、40²）個の文字を表現できる。

左→左	右→左
左→右	右→右

漢字コード表の全体は４つに区切られていて、それぞれの位置を左に示すように、キーボードの入力順序で表現する。たとえば、左下の部分（図の中の網掛け部分）ならば、左手キーを押したあと、右手キーを押すことになる。

キーボード

両手にそれぞれ20個のキーがある。このキーが漢字コード表のマトリクスに対応する。

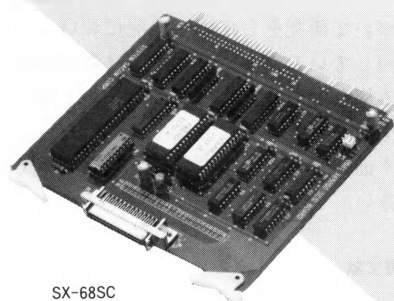
入力例：漢字コード表の「N」をキー入力する

１）まず、「N」を示すマスがあるマトリクスが、漢字コード表全体のうち、どの位置にあるかをみる。
右上部分にあるので、押すキーは右手キー→左手キーの順である。

２）その右上部分のなかでの、マトリクスの位置をみる。
４列２段目である。１)で確認したように最初に押すキーは右手キーなので、①キーを押す。

３）マトリクスのなかの「N」の位置をみる。
４列３段目である。１)で確認したように２番目に押すキーは左手キーなので、②キーを押す。

NEW PRODUCTS

X68000用SCSIインタフェイスボード SX-68SC システムサコム

SX-68SC

システムサコムでは，X68000用SCSIインタフェイスボード「SX-68SC」を1月中旬より発売する。

「SX-68SC」は，純正品「CZ-6BS1」とハード的にまったく同じ動作を行えるもので，添付のSCSIドライバもシャープより供給され，OS上での認識は純正品とまったく同様である。純正品と違い接続コネクタにハーフピッチの50ピンを採用している。

主な仕様は以下のとおり。

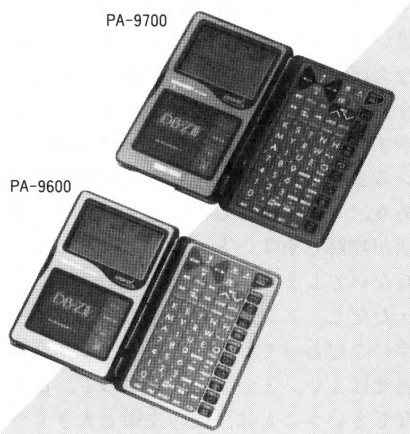
・コントローラ　MB89352
・信号伝送方式　不平衡型(シングルエンド)
・転送モード　非同期転送
・コネクタ　ハーフピッチ50ピン

価格は，26,800円（税別）となっている。

<問い合わせ先>
㈱システムサコム　☎03(3635)5145

ハイパー電子システム手帳 PA-9700/9600 シャープ

シャープでは，ハイパー電子システム手帳の新しいシリーズとして「PA-9700」「PA-9600」を発売した。

PA-9700
PA-9600

本機は，従来あったハイパー電子手帳シリーズに，画面に表示する情報量を65%アップし，タイムマネージメントをサポートするため，仕事に優先順位をつけるアクションリスト機能などの強化を行ったものである。

表示画面に12×12ドット文字を使用することで，16文字×10行の画面表示を実現している。これによって，週間スケジュールでは，一度に14件まで確認することができ，一覧性が向上した。また，手書きメモ機能のエディットツールも強化されている。約4万語の国語辞典も内蔵しており，意味，用例，熟語などのほか，JIS，区点，シフトJISコードでの検索も可能である。さらに，音訓や総画数などからも検索できるようになっている。

価格は，「PA-9700(容量256Kバイト)」が59,000円，「PA-9600(容量128Kバイト)」が48,000円（ともに税別）である。

<問い合わせ先>
シャープ㈱　☎06(621)1221,043(299)8210

ICカード3種類 PA-9C61/3C50S/2S シャープ，講談社，主婦と生活社

シャープ，講談社，主婦と生活社より，シャープ電子手帳用ICカード3種類が発売された。

PA-9C61
PA-3C50S
PA-3C52S

●浅野八郎の占い四柱推命学カード〈PA-9C61〉

シャープは，「浅野八郎の占い四柱推命学カード」を発売した。このカードは1988年3月に発売した「四柱推命学カード」を浅野八郎監修のもとで，さらに多彩な占いを実現したものである。

利用者は，生年月日と性別を入力するだけで，簡単に四柱推命占いを行える。「性格」「健康」など6項目の占い，「恋愛，結婚」「仕事」の相性占い，四柱推命の基本となる運命式の表示（PA-7000では不可）などが行える。

また，専門用語55語の用語辞典も収録している。価格は7,000円(税別)。

●血液型＋星占いカード〈PA-3C50S〉

講談社では20歳前後の男性をターゲットにした「血液型＋星占いカード」を発売した。

このカードは，利用者の星座と血液型から，基本運，日運，相性運，交際術の4つのジャンルで占いができる。基本運は，9分野（運命，性格，愛情，対人関係など）を占うもので，日運は2020年までその日ごとのラッキー運が7分野で占える。

相性運は気になる彼女との相性，攻略法を占え，交際術は上司，先輩など8パター

ンの相手を対象にアドバイスを得られるような構成となっている。価格は8,500円(税別)。

●ビジネスの知識百科カード〈PA-3C52S〉

主婦と生活社では，すぐに役立つビジネス情報を収録した「ビジネスの知識百科カード」を発売した。

カードには，ビジネスに関するさまざまなルールを「ビジネスマナーの常識」「冠婚葬祭のマナー」「ビジネス文書の知識」「ビジネス法知識」「ビジネス豆知識」と大きく5つのカテゴリーに分類しており，説明事例は950件ある。

さらに，「ビジネス便利帳」「生活便利帳」には，官公庁，大使館，全国の主要ホテルなどの施設，組織の電話番号をそれぞれジャンル別，地域別に約1,260件収録している。検索は，メニュー検索によって行われ，それぞれのカテゴリーの中で細分類したデータを引き出すことができる。価格は7,000円(税別)。

〈問い合わせ先〉

シャープ㈱　☎06(621)1221,043(299)8210
㈱講談社　☎03(5395)3434
㈱主婦と生活社　☎03(3563)5228

FORTHコンパイラ
MF-68K/OS-9
マイクロフォース

マイクロフォースは，X68000用FORTHコンパイラ「MF-68K/OS-9」を発売した。

「MF68K/OS-9」は，いままで発売されていた「MF-68K」をOS-9に対応させたものである。そのため，新たに標準のOS-9ライブラリとしてユーザーステートシステムコール，I/Oシステムコール，イベント，Mathに対応した浮動小数点ライブラリを装備，また，シャープX68000用OS-9のライブラリも装備している。

なお，Human68kバージョンの「MF-68K」とソースレベルで互換性を保っている(グラフィックライブラリなどで一部仕様の違うものがある)。

価格は19,800円(税別)，バージョンアップは3,300円(製品，送料込み)となっている。

〈問い合わせ先〉

マイクロフォース㈱　☎03(3756)1988

INFORMATION

音と音楽のスーパーイベント
ローランド・サウンド・パーティ
ローランド，ローランドMCクラブ

ローランドとローランドMCクラブでは，一般ユーザー向けのイベント「ローランド・サウンド・パーティ」を1993年2月11日(木，祝日)新宿ルミネホールACTにて開催する。

「ローランド・サウンド・パーティ」では，新製品の発表試奏会や著名プレイヤーによる演奏やステージが用意されている。当日は，先着100名に特製トレーナーをプレゼント，さらに来場者の中から抽選によりJV-30が当たるイベントも用意される。

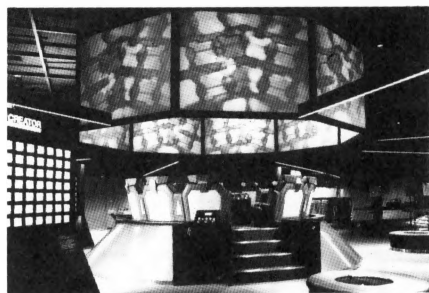
ステージでは，「ムーンライダーズ」の白井良明氏と「すかんち」のローリー寺西氏によるトークライブ，そしてホッピー神山ユニット「FINAL:930211」によるスペシャルライブも行われる予定だ。

入場は無料。都内及び近郊の有名楽器店，パソコンショップで詳細を記載したチラシ，入場券を準備している。

〈問い合わせ先〉

ローランドMCクラブ　☎03(3251)2833

ハイテクを使った未来映像ホール
シャープハイテクノロジーホール
シャープ

シャープは，千葉幕張新都心に新しく完成した「シャープ幕張ビル」内に，最先端の技術を生かした新しい装置やツールを使って，“技術と文明の進歩と未来”を楽しく勉強したり体験できる“ハイテクノロジーホール”を開設した。

ハイテクノロジーホール

本ホールは，7つのコーナーで構成されており，それぞれ場内いたるところに設置された液晶画面によって，新しい映像感覚を体験できるようになっている。

コーナーの内容は以下のとおり。

1)　フライングユートピア

17世紀の哲学者“フランシス・ベーコン”が400年前に思い描いた，未来世界を再現する。座席が昇降，回転しながら100型液晶ビジョン9台による360度の全周マルチ画面を楽しめる。

2)　インフォトリックス

140枚の液晶ディスプレイを使った映像案内板を場内4カ所に配置したもの。合計560枚の液晶に今世紀の文化，社会などの映像が縦横に走る。

3)　ハイパーナビゲーター

“ベーコン”が予言した未来技術が，今日の暮らしに結びつくさまを82の技術領域から自由に学ぶことができる。

4)　ハイパークリエーター

自分の顔を撮り込んでイラストを書き加えるなど，楽しみながらCGを体験できる。

5)　シャープトゥデイ

ハイビジョン，液晶TVなどの映像機器からFAX，ワープロ，パソコンなど情報機器まで自社最新製品を展示。自由に体験することができる。

6)　シャープトゥモロウ

半導体レーザーやCCD，LEDなど身近にある“オプトエレクトロニクス”の最新技術やその応用を，簡単な実験を通じて興味深く学べる。

7)　液晶ハイビジョンシアター

液晶ハイビジョン3台(360万画素)を駆使し，細密で鮮やかな220型の大画面を満喫できる(51席)。

開館時間は，月曜日から金曜日の10:00から17:00まで，土曜・日曜・祝日は休館となっている。入場は無料。

FILES Oh!X

このインデックスは，タイトル，注記　著者名，誌名，月号，ページで構成されています。2月といえば「豆まき」。でも追い出された鬼はどうするんでしょうね。みんなで一緒にこたつでぬくぬくするのもいいかもね。

参考文献
I/O　工学社
ASCII　アスキー
月刊PC　ソフトバンク
コンプティーク　角川書店
C Magazine　ソフトバンク
テクノポリス　徳間書店
POPCOM　小学館
マイコンBASIC Magazine　電波新聞社
My Computer Magazine　電波新聞社
LOGIN　アスキー

一般

▶アルゴリズムを見切ったぞ!?
CG集を作るために。各種グラフィックツールで描いたデータを表示させるプログラムを機種ごとに紹介。——おにおん，テクノポリス，1月号，144-148pp.

▶新製品Short Cut
X68000用サブCPUボード「POLYPHON」，高校・専門学校向けポケコン「PC-G805/815」，SX-WINDOW対応FM音源音色作成ツール「SOUND SX-68K」など新製品を紹介。——編集部，マイコンBASIC Magazine，71-75pp.

▶ワープロ/パソコン通信新聞
スキー情報，EYE-NET「平成教育委員会」連動企画など大手ネットの最新情報，短期連載・パソコン通信への道，草の根ネット紹介など。——山本まさこ，マイコンBASIC Magazine，1月号，78-82pp.

▶BASICプログラミング講座　第9回
BASICを使い，簡単な方程式をグラフ化して解いてみよう。数学がわかりやすく身近になる！——東　幸太，マイコンBASIC Magazine，1月号，90-94pp.

▶年末ハード買い占め大作戦
ゲームソフトの価格と性能比で現行のパソコンを比較。購入の参考に。——編集部，LOGIN，23号，251-271pp.

▶MUSIC LABO
コンピュータミュージックに挑戦する人のための連載。今回はMIDI環境やそれを操るソフト，インストールなどを解説。新製品紹介はヤマハ「QY-20」など。——編集部，LOGIN，23号，280-287pp.

▶NEW MACHINE TEST!!
1992年秋に発売された各社の新製品をレビュー。マルチメディア対応機98MULTI，Macintosh IIvx/viなどが登場。——編集部，ASCII，1月号，205-229pp.

▶Mathmaticaは科学の枠組みを変える
Mathematica開発の中心人物であり，高エネルギー物理や宇宙論の分野で活躍しているスティーブン・ウルフラム氏にコンピュータと物理学の問題についてインタビュー。——岡本賢吾，ASCII，1月号，230-236pp.

▶CD-ROMの世界へ，ようこそ
CD-ROMでパソコンライフを広げたい人へのガイド。ドライブやソフトの紹介，オリジナルCD-ROMの作り方など。——志村拓ほか，ASCII，1月号，254-276pp.

▶バカパパのモノを買い物
フロッピー郵送用の封筒，ディスプレイ拡大フィルタなどちょっとイロモノなパソコングッズを紹介。——バカパパ，ASCII，1月号，354-355pp.

▶ラッキー！ハッピー！オッケー！
パソコン通信のチャットのログや会議室のメッセージの転載の著作権について，弁護士の山下氏に聞く。——編集部，ASCII，1月号，376p.

▶パソコンマーケットガイド
各種ソフトや，プリンタバッファ，モデムなど周辺機器と，全国パソコンショップをコメントつきで紹介。——編集部，My Computer Magazine，1月号，44-113pp.

▶マイコンからMy Computerへ
創刊15周年記念企画の第5回。今月はネットウェアの先駆者ノベルの渡辺和也社長へのインタビュー。——編集部，My Computer Magazine　1月号，134-143pp.

▶CGの仕掛人
コンピュータグラフィックの最前線をルポ。今月はその先駆者といえるNHKのCGへの取り組みを紹介。——大窪志保，My Computer Magazine，1月号，146-149pp.

▶フロプティカル・ディスクが大容量な理由
従来のフロッピーとの互換性を保ち，21Mバイトと大容量のフロプティカルディスク。その仕組みを探る。——編集部，My Computer Magazine，1月号，168-171pp.

▶防磁ケースのフロッピーは本当に安全か
磁気や衝撃，ホコリからの防御性が売り物のフロッピーケース。その本当の実力を磁気センサーで検証。——石川至知，My Computer Magazine，1月号，178-182pp.

▶速報！ COMDEX '92 FALL
世界最大のパーソナルコンピュータ展示会・秋期コムデックスの模様を伝える。大手メーカーの展示内容やマルチメディアをめぐる動きなどの詳細レポート。——編集部，My Computer Magazine，1月号，232-235pp.

▶星はTEPIAでさがせ！
東京・北青山のTEPIAで開催中の「HUMAN in SPACE」を紹介。火星探査のバーチャルリアリティからペンシルロケットまで，宇宙開発の過去未来を学べる——編集部，My Computer Magazine，1月号，246-249pp.

▶マイコン考古学
「パソコンサンデー」のドクター宮永によるマイコン史。Cの登場から現在のブームの原因を考察する。——宮永好道，My Computer Magazine，1月号，286-289pp.

▶なんでもQ＆A
書院パソコンのパレット設定方法，中に入れる単3電池は何に使われているか，などの質問に回答。——シャープ，My Computer Magazine，1月号，306-307pp.

▶I/O目安箱
「ハイテク時代の「素朴な疑問」」と称し，技術的でない質問に，私情混じりで回答。パソコンから出る電磁波とは？ など。——KANAMI，I/O，1月号，73-76pp.

▶マルチメディア'92
11月・幕張メッセでの「マルチメディア'92」。CD-ROMと32ビットCPU搭載のマルチメディア指向マシンが登場，ソフトなども盛況。——編集部，I/O，1月号，90-91pp.

▶「常温核融合」に新しい光
NTT基礎研究所が「真空法」により常温核融合に新たな可能性を開いた。その原理とニュースの意義について。——編集部K^2，I/O，1月号，112-113pp.

▶スーパーコンピューティング入門
数学パズルの古典「ライフゲーム」。ルール解説に始まり，発見されたパターン，オートマトンと呼ばれる概念も解説。——林智雄，I/O，1月号，139-141pp.

MZシリーズ

MZ-700/1500(S-BASIC)

▶紫と緑の理論
キャラクターを上下左右に動かし，スペースキーで魔法を使う。紫を引けば緑が押されるふしぎなパズルゲーム。——ナルサス，マイコンBASIC Magazine，1月号，112-113pp.

X1/turbo/Z

X1シリーズ

▶STRATEGY OF TRIUMPH
相手の持ち機を全滅させれば勝ち。2人用対戦シミュレーションゲーム。——岩崎雄大，マイコンBASIC Magazine，1月号，136-137pp.

▶石遍路
アステロイドを走り抜けろ！　避けて撃つシューティングアクションゲーム。——BANCO，マイコンBASIC Magazine，1月号，138-139pp.

▶F-ZERO　～BIG BLUE～
スーパーファミコン用ゲームよりミュージックプログラム。要FM音源ボード(NEW-FM音源ドライバ)。——川村賢治，マイコンBASIC Magazine，1月号，157-158pp.

X68000

▶GAMING WORLD
ド派手アクションゲーム「ストライダー飛竜」，ズーム初のF1レースアクション「オーバーテイク」，かわいいキャラクターたちが冒険「エトワールプリンセス」。年末年始に発売予定の各機種ゲームソフトカタログも掲載。——編集部，テクノポリス，1月号，24-29，38-50pp.

▶SOFT EXPRESS
小説・アニメでおなじみ「ロードス島戦記II」ついに移植。——編集部，コンプティーク，1月号，62p.

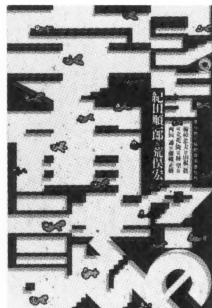
▶HOW TO WIN
カプコンのアーケードゲームからの移植「ストライダー飛竜」を攻略。X68000の能力を最大限に生かした完全移植だ。ほかに「バーンウェルト」。——編集部，コンプティーク，1月号，134-137 152-153pp.

▶Software Hot Press
過去の名作アーケードゲームがX68000で復活！「ムー

ンクレスタ/テラクレスタ」。1枚のディスクで低価格。お得なソフトだ。――編集部，POPCOM，1月号，24p.
▶ゲームの達人
車載カメラアングルで君もF1ドライバーになりきろう「オーバーテイク」。飛竜のアクロバットアクションで地球を救え！「ストライダー飛竜」。――編集部，POPCOM，1月号，98-99，104-105pp.
▶Submarine
潜水，魚雷を駆使して敵艦隊をかわし，輸送船団の情報をつかみ撃沈する。本格的な潜水艦シミュレーション。――立石隆二，マイコンBASIC Magazine，1月号，140-142pp.
▶素潜り
サメに食われないように貝を拾う。貝を捨てれば速く浮き上がれるけど点数は少ないぞ。――渋谷正徳，マイコンBASIC Magazine，1月号，143-145pp.
▶幽体離脱
オバケになった影さんは無事人間に戻れるか。グラサンかけたオバケをゴールへ導くワンキーゲーム。――加藤淳一，マイコンBASIC Magazine，1月号，146-148pp.
▶ドラゴンセイバー　～ウルティマ～
ナムコのアーケードゲームよりミュージックプログラム。要NAGDRV，CM-64。――上田浩司，マイコンBASIC Magazine，1月号，159-161pp.
▶オーバーテイク
F1ファンも満足のこだわりの一作というズームの「オーバーテイク」。コース一覧や攻略法を掲載。――板場利光，マイコンBASIC Magazine，1月号，210-213pp.
▶現行ゲームソフト・カタログシリーズ
市販されているゲームソフト一覧。第1回はX68000のソフト359点を掲載。ソフト選びに役立つかな？――編集部，マイコンBASIC Magazine，1月号，214-226pp.
▶NEW SOFT
オープニングから超カワイイ！　ロールプレイングゲーム「エトワールプリンセス」。細かい演出も必見。――編集部，LOGIN，23号，28p.
▶最新ゲーム徹底解剖!!
来期に向けて南アフリカ攻略だ！「オーバーテイク」。――編集部，LOGIN，23号，178-181pp.
▶X68000新聞
X68000用サブCPUボード「POLYPHON」。新着ソフト「ムーンクレスタ/テラクレスタ」「究極タイガー」。C言語講座の第6回は「プログラムの仕組み」。――編集部，LOGIN，23号，290-293pp.
▶NEW SOFT
なつかしのアーケードゲーム移植版「テラクレスタ/ムーンクレスタ」。――編集部，LOGIN，24号，32p.
▶最新ゲーム徹底解剖!!
まだまだしゃぶりつくすぞ！「三國志III」のゲーム画面のお城を紹介。新作「オーバーテイク」はマシンのセッティング研究。――編集部，LOGIN，24号，178-185pp.
▶X68000新聞
「ストライダー飛竜」「キングス・ダンジョン」「SOUND SX-68K」。C言語講座第7回は「printf関数」。――編集部，LOGIN，24号，264-267pp.
▶AV STRASSE
豊富なエフェクトと自在なタッチが魅力のグラフィックソフト「MATIER」の使用感をレポート。――編集部，ASCII，1月号，309-312pp.
▶FREE SOFTWARE INDEX
大手通信ネットにアップロードされたソフトの紹介ページ。X68000用バッチファイル支援ソフトBATX.x，SX-WINDOW用動画・静止画再生プログラムMovie.xなど。――編集部，ASCII，1月号，387-393pp.
▶長期ロードテスト
X68000EXPERT IIの近況報告。TeXを導入するも，出番がなくあまり習熟せず，便利な通信端末として活用しているとの由。――編集部，ASCII，1月号，397-407pp.
▶なんでもQ＆A
SX-WINDOWの外字作成にパターンエディタを使うには？ SX-WINDOW開発キットのサンプルの内容は？ の2問に回答。――シャープAVCシステム事業推進室，My Computer Magazine，1月号，304-305pp.
▶SLG Laboratory
「三國志III」の仕上がりプレイ。――猪野清秀，My Computer Magazine，1月号，360-363pp.
▶GAME BOX
デモにこり，データベースなども揃えた力作「オーバーテイク」。――竹沢ながせ，I/O，1月号，82-83pp.
▶More Reviews
「POLYPHON」を紹介。東芝製のTMP68303に増設RAM，MIDIインタフェイスなどを組み込んだ使いでのあるサブCPUボードだ。――牟田拓，月刊PC，1月号，296p.
▶GCCで学ぶX68ゲームプログラミング
先月に続きG＋＋スプライト奮闘記。前回の問題点を改善。――吉野智興，C Magazine，1月号，138-142pp.

ポケコン

PC-E500
▶Triangle
三角形のブロックを組み合わせ，四角形にすればブロックが消えるパズルゲーム。――有光喜一，マイコンBASIC Magazine，1月号，149-150pp.
▶SPHINX2
他人を出し抜き，いち早く宝の部屋へ！　マルチプレイヤーゲーム。――春田秀樹，マイコンBASIC Magazine 1月号，151-152pp.

新刊書案内

コンピューターの宇宙誌
紀田順一郎，荒俣　宏　他著
ジャストシステム刊
A5判　266ページ
2,000円(税込)

ジャストシステムというと「一太郎」やPC-9801というイメージが強いが，それは杞憂もいいところのなかなか面白い本であった。本書に収められているのは荒俣宏と紀田順一郎の2人がコンピュータを創造的活動に使っている人たちと対談してまわった結果である。コンピュータをどう使うか，コンピュータが何をもたらしたか（あるいはもたらすか）といった点を中心にごりごりと攻めていくさまは，対談集の密度を非常に濃いものにしている。対談相手は梅棹忠夫，山根一眞，立花隆，林望，西垣通，藤幡正樹といったそうそうたる人々であり，それぞれが自分の専門分野に立脚したテーマで語りあう。CGアーチストの藤幡正樹氏を除くと，多くがパソコン黎明期に日本語とコンピュータの親和に苦労していたり，資料のデータベース化に苦労しており，そのほとんどは今でも解決されていないという点を直接/間接に語っている。「知的生産」というとうさん臭いけれども，これだけ一流どころが揃っていると奥が深い。
6人の中でも特に新鮮だったのは，この対談集をつくるきっかけになった国立民族博物館館長の梅棹忠夫氏と，書誌学という馴染みの薄い学問の専門家でありエッセイストでもある林望氏のページだ。特に梅棹氏との対談には，ほかの人の倍以上の分量を割いており，そこで語られるコンピュータに対する視点の確かさに驚かされる。コンピュータがなければ処理できない情報を大量に抱えているのに，日本語がネックになってその整理がうまくいかない。結果として浮き上がった問題はコンピュータ側ではなく，文明論的に語られない日本語や，インデックスの概念がまったく発達しなかったために招いてしまった野放しの日本語側にあった。本書の面白さはあくまでも「日本」という土壌にこだわったところにあり，それがほかのコンピュータ関係対談集との差異を際立たせているといえよう。　(K)

知って得する
ソフトウェア特許・著作権
古谷栄男/松下　正/真島宏明共著
アスキー出版局刊
☎03(3486)7111
A5判　295ページ
2,500円(税込)

「著作権」「知的財産権」という言葉は，最近ひんぱんに問題にされている。しかし，なにしろはっきりとした形のないものだけに，はなはだあいまいで，知らず知らずのうちに侵害していたということは容易に起こり得る。
本書では具体例などを挙げて，権利とは，主張とはどういうもので，どう対処すべきかが示されている。それらは他人を侵害しない，また侵害されないためにもぜひ知っておくべきことである。
今後，より重要になっていくであろう「知的財産権」について正しく認識することは，その作者・開発者に対する礼儀の第一歩ではないだろうか。

電卓で遊ぶ数学
大野　栄一著
講談社刊
ブルーバックス941
☎03(5395)3524
新書判　264ページ
780円(税込)

10桁電卓でどこまで高度な計算ができるか。本書では，電卓の最低限度の機能を想定して，数学を学びながら，電卓を使いこなすさまざまなテクニックが紹介されている。
キーの説明に始まり，累乗計算，逆数計算，16進数の計算，n乗根の計算，循環小数，連立方程式，多角形の面積……と計算は続いていく。もちろんすべて小さな電卓で。「実践数学編」の章では，ローンの返済計算や偏差値計算も登場する。
裏表紙には「キミは電卓でパソコンに勝てる！」と書いてある。それが本当かどうかを知るには，実際に電卓片手に本書を読んでみるしかない。

X-OVER・NIGHT
(クロスオーバーナイト)

[第31話] '93年電子的生活環境予測

TAKAHARA HIDEKI 高原 秀己

今回は，われわれを取り巻くエレクトロニクスの身近なレベルでの環境に関して，今年1993年のストリームを予想してみた。情報に基づいた分析の結果のものから，まったくの妄想をベースにしたものまでさまざま並べてある。サラサラっと読み流していただければ幸いである。

●CD

チャゲ&飛鳥，ドリカムなどメガヒットが相次ぎ，音楽ソフト会社が隆盛だと指摘された'92年。ある新聞では「CDがシングルならハンバーガー2個の感覚で売れるようになったから」などと分析していた。

この分析は明らかに間違い。CDが飛ぶように売れるようになったのは，CDプレイヤー内蔵ラジカセや車載型CDプレイヤーが当たり前になり，「ディスクマン」などの携帯プレイヤーが定着したからだ。手軽に誰でもCDを聴く環境が整ったのは，ここ2年のこと。

'93年はこの傾向がますます強まるだろう。あわせて再販商品の値下げが増えるから，価格競争も起こる。一方では，会社の壁を越えた複数アーティストの混載ソフトなども出てくるのではないだろうか。ソフトの買い方だけではなく，「商品」としての意識のされ方も「レコード」の時代とは違う。

●BS/CS

BSは，WOWOW日本衛星放送が累積赤字400億円を抱えて倒産寸前。回転資金も底をついている。

打開策はないといわれているが，事実なさそうだ。結局，いくつかの銀行が郵政省に頼まれて支援団体を作り，どこかの大手資本が主体になって再建するしかないのだが，引き受け手はよほど酔狂な会社ということになる。

一方のCSに関しては，もうお先真っ暗としかいいようがない。番組内容はといえば，あちこちのケーブルTV局で流しているものを，ほぼそのまま流すだけ。それでいてパラボラアンテナ(BSアンテナとは兼用できない！)と専用デコーダを合計20万円出して，さらに月間視聴料金を払って見てくれという。BSがようやく普及しつつある段階で，こんなものに大金を払う人などそうそういるはずがない。会社は当然ペイせず，大赤字に苦しんで内容は低下，ユーザーは逃げる，というサイクルが手に取るように予想されてしまう。とにもかくにも，たとえタダでも，これ以上アンテナを設置したり，デコーダを置きたくないという状況なのである。

その逆に，自治体ごとにパラパラとできているケーブルTV局には，かなり期待できる。BSまで全部ひっくるめて頭金10万円程度というのは，はるかに魅力的だからだ。ケーブルTV局によって格差はすごく出るはずだが，なかには成功するところがきっと出てくる。

●マルチメディア

絵に描いたモチは，まだまだ実用段階には至らない。ただし「マルチメディア」の前段階として，「デュアルメディア」なんかは意識されるようになるだろう。すでにX68000なんかはこの域にあるといえばあるのだが，もう一歩進んだ使い方が出てくるということだ。この場合，なにも映像なり動画にかぎらないことに注意したい。パソコン通信と音楽なんていう軽いメディアミックスでいいのだから。いずれにしても，ソフトというか，利用目的を確立させることが重要だ。

●パソコン

DOS/Vによる低価格IBM互換機が話題を呼んだのが'92年の特徴だが，実際に商品が売れて話題になったのではない点にわれわれは注意したい。中古市場でDynaBookが山のように捨て値で売っている現実をあわせて考えると，決してDOS/V互換機の見通しは甘くない。

少なくとも'92年にPC-9801の対抗機として売れた商品は，Macintoshだけだったのである。

Macintoshは，買ってすぐ使える雰囲気がある。しかしDOS/Vマシンは，周辺機器を買い揃えてソフトを集める段階が意外に大変なのだから，マニアでも骨が折れる。一方で，ユーザー層は裾野方向に拡大している。ワープロより難しい機械は使えない人が主体になるのだ。

結論。DOS/Vマシンは話題ほど市民権を得ることができない。ただし，NECが値下げを一切しないのならば，話はやや変わってくるのだが。

●X68000

そろそろラップトップが発売されていい時期にきているはず。

いうまでもなく，カラー液晶が必要だ。だが，ディスプレイパネルを取り替え式にしておいて，いずれ交換することが可能，という方法でもいいだろう。さらに液晶テレビ機能も追加してくれれば，いうことはない。

一方では「BS/CS対応X68000」とか「ハイビジョンX68000」なんてのも高級タイプ機の一環として出てきてもいいかもしれない。

●ブライダルの「ブーム」

おまけ。

5月28日の貴花田ー宮沢りえの結婚式を機に，下火だったブライダルが盛り上がるはず，という説がある。一方では「年明けにも皇太子殿下のお相手が発表されるのでは？」などという推測もある。

芸能スポーツ分野で最強の組み合わせの結婚があり，さらにロイヤル・ウェディングなどということになれば，確かにこりゃ大変だ。一方では雑誌も目玉特集のネタが不足しているから，絶対にあおる。物件が動かない不動産業界にとっても，結婚が増えれば確実に需要が高まる。企業パーティが不足しているホテルも大歓迎。結婚ブームにでもなれば，喜ぶ人は無限にいる。

こうなると，ブームに弱い日本人。「結婚しない症候群」だった女性も一気に宗旨変えしても決しておかしくない。

というわけで，不本意な独身である男性諸氏は期待しましょう。

●編集部へのメッセージ

今月号の特集について

いちばん良かった記事	これから載せてほしい記事内容
Oh!Xゲーム大賞	ゲーム大賞推薦理由
1.	推薦理由（　　）
2.	
3.	
4.	

あなたの愛機は(所有機種に○印をつけてください)　　　　ない
X1(マニアタイプ,C,D,F,G,twin) X1 turbo(model 10,20,30,40,II,III,Z,ZII,ZIII)
MZ-(80K/C, 1200, 700, 1500, 80B, 2000, 2200, 2500, 2861)
X68000(初代,ACE,PRO,PROII,EXPERT,EXPERTII,SUPER,XVI,Compact, HD)
その他　　　　MIDI楽器(　　　　)
FD(　　基)　TAPE　QD　HD(　　MB)　MO　プリンタ(　　　　)

年齢　　歳　　パソコン歴　　年　　男・女　　プレゼントNo.

振替用紙

←点線から、きれいに切り取ってご使用ねがいます。

払込票

通常払込料金
加入者負担

口座番号	東京	1	29307
加入者名	ソフトバンク株式会社		
金額	億 千 百 十 万 千 百 十 円		
払込人住所氏名	※		
備考		受付局日付印	

切り取らないで郵便局にお出しください。

切り取り線

払込通知票

通常払込料金
加入者負担

口座番号	東京	1	29307	金額	億 千 百 十 万 千 百 十 円
加入者名	ソフトバンク株式会社			料金	払込み / 特殊 円
払込人住所氏名	※ （郵便番号　　　）			備考	
				受付局日付印	

記載事項を訂正した場合は、その箇所に訂正印を押してください。

各票の※印欄は、払込人において記載してください。

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

切り取り線

【定期購読のご案内】

●定期購読のお申し込みは、この郵便振替用紙のみとさせて頂き、銀行振込・現金書留によるご入金は、ご遠慮下さい。

●受付締切は、

1日発売　　：　発売日前月10日振込
8日発売　　：　　〃　　15日振込
15・18日発売：　　〃　　25日振込
です。

〈例〉　4月1日発売（Oh！PC 4月15日号）の場合、お振込の締切は3月10日です。
締切に間に合わなかった場合は、自動的に次号からの発送となります。
なお、すでに発売されているもの、また、お振込が締切に間に合わなかった月号のものは、定期購読ではお求めになれません。書店でご購入ください。

●定期購読誌のお届けは書店発売日より遅くなりますのでご了承下さい。

「発売日一覧」

◇毎月1・15日発売
Oh！PC

◇毎月18日発売
Oh！X
Oh！FM TOWNS
C MAGAZINE
Oh！Dyna

◇毎月8日発売
月刊情報処理試験
LANTIMES
THE WINDOWS
DOS／Vmagazine
UNIX USER

●月刊情報処理試験は93年1月号より定期購読料金を改訂させていただきます。お申し込みの際はご注意下さい。

切り取り線

切り取らないで郵便局にお出しください。

この欄は、加入者あての通信にお使いください。

送り先

お名前　フリガナ　　お電話

ご住所　フリガナ
〒

定期購読申込書

□Oh！PC	年間	（23回）	12,880円	（新規／継続 NO.　　）
□Oh！PC	6ヶ月	（12回）	6,720円	（新規／継続 NO.　　）
□Oh！X	年間	（12回）	7,200円	（新規／継続 NO.　　）
□Oh！FM TOWNS	年間	（12回）	7,440円	（新規／継続 NO.　　）
□C MAGAZINE	年間	（12回）	11,760円	（新規／継続 NO.　　）
□Oh！Dyna	年間	（12回）	9,120円	（新規／継続 NO.　　）
□月刊情報処理試験	年間	（12回）	9,360円	（新規／継続 NO.　　）
□月刊情報処理試験	6ヶ月	（6回）	4,680円	（新規／継続 NO.　　）
□LANTIMES	年間	（12回）	17,760円	（新規／継続 NO.　　）
□THE WINDOWS	年間	（12回）	11,760円	（新規／継続 NO.　　）
□DOS／Vmagazine	年間	（12回）	9,360円	（新規／継続 NO.　　）
□UNIX USER	年間	（12回）	11,760円	（新規／継続 NO.　　）

〔備考欄〕

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。　（郵　政　省）

愛読者プレゼント

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1993年２月18日の到着分までとします。当選者の発表は1993年４月号で行います。また，雑誌公正競争規約の定めにより，当選された方はこの号の他の懸賞には当選できない場合がありますのでご了承ください。

ハミングバードソフト ☎06(315)8255

1 ロードス島戦記II —五色の魔竜—

X68000用
5″2HD版

9,800円(税別)

3名

イベントたくさん，アイテムどっさり，魔法もいっぱい，と本当に骨の髄までしゃぶれそうなロールプレイングゲーム。もちろん本筋のほうも面白いし，戦闘はタクティカルコンバットだ。

電机本舗 ☎03(3447)1773

2 XIN/OUT II ver.7.0f

X68000用 3.5/5″2HD版

14,800円(税別) 2名

この「XIN/OUTII」を使うと，MacintoshとX68000の間でファイルのやりとりができるのはご承知のとおり。これなら巨大なファイルも転送可能。今回はSystem7に対応した。

3 上昇気流 vol.4

10名

もう４回目なのね。すでに毎年恒例となりつつある，高橋哲史君の「上昇気流」プレゼント。今回も身を削るようにして制作したそうなので，機会があれば買ってあげてちょうだい。

4 1993年 卓上カレンダー

Aイマジニア
4名

Bソフトバンク
20名

ちょっと遅ればせながらという感もありますが，イマジニアさんとソフトバンクの1993年のカレンダーをプレゼント。今年も１年がんばりましょうってとこかな。

12月号モニタ当選者

1 １Mバイト増設RAMボード （群馬県)原田進 2 ２Mバイト増設RAMボード （埼玉県)迫田勝弘 3 ２Mバイト増設RAMボード （宮城県)阿部勝久 4 ２Mバイト増設RAMボード （大阪府)畑中英喜 5 数値演算プロセッサボード （高知県)井上達雄 6 数値演算プロセッサ （東京都)藤田瑞穂 7 MIDIボード （東京都)乗本貴史 8 増設５インチFDD (群馬県)黒澤典義 9 システムラック （東京都)佐藤弘憲 10 サイバースティック （埼玉県)岡田具明 11 CARD PRO-68K ver.2.0 （北海道)新井誠治 （東京都)大内泰一 （神奈川県)八木明 （兵庫県)多田哲也 （岡山県)小野智章 12 EasyPaint SX-68K （千葉県)北久保晴康 （埼玉県)小林裕二 （和歌山県)河本直規 （広島県)上村光治 （山口県)大野貴志 13 グラフィックライブラリ vol.3 （埼玉県)石本ヨゼフ （東京都)村澤博人 （神奈川県)柴田寿 浅井和彦 （静岡県)秋野潤 14 ダウンタウン熱血物語 （長野県)大槻尚義 （東京都)河野太郎 （大阪府)宮永宏樹 （岡山県)三宅良和 （福岡県)山口裕二 15 熱血高校ドッジボール部サッカー編 （宮城県)酒井弘志 （愛知県)出口賢次 （兵庫県)井上卓顕 （岡山県)寺尾文治 （福岡県)諸藤健一 16 フロッピーディスクケースＡ （宮城県)鈴木政宏 (福島県)伊藤直広 （茨城県)菅野宗 （千葉県)浮田衛 西村宏功 松戸康行 （埼玉県)高橋勝 （東京都)五十嵐正治 江村勝彦 古賀宏昭 鈴木陽二郎 増田秀樹 （神奈川県)小島靖幸 鈴木康之 藤本格 古川博一 由岐中康司 （富山県)清河豊 （岐阜県)井戸直樹 （愛知県)鈴木健児 干場修二 （大阪府)石田貴志 須賀院隆志 村上剛規 （兵庫県)秋定貴文 田間豊常 （岡山県)谷弘志 （香川県)高尾明宏 （愛媛県)小濱英司 富永博之 Ｂ （埼玉県)根本敬四郎 （愛知県)早川博 安井太郎 （兵庫県)村瀬正美 （広島県)秋山欣之 17 中華大仙 （千葉県)浜田研一 （東京都)尾形敦 （神奈川県)畑野淳嗣 （三重県)増川一詞 (滋賀県)小西拓馬 （敬称略）

以上の方々が当選しました。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。

QUESTION and ANSWER

IOCSコールを使用しないでAD PCMを鳴らす方法を教えてください。　富山県　砂原　雅人

ADPCMによる音声の再生は，AD PCMデータレジスタにAD PCM形式のデータを連続的に書き込むことで行われます。AD PCMデータは圧縮されているとはいえ，転送するデータが大きければCPUにかかる負担も大きくなります。X68000ではDMACのチャンネル3をAD PCMへのデータ転送専用に割り当てて，データ転送の高速化とCPUの負担を減らす設計になっています。

IOCSコールを使わずにAD PCMから音声を出力するには，音声の再生レート，パンポット(出力チャンネル)，DMACによる転送に必要なパラメータを設定したあと，転送開始コマンドを送信します。

再生レートはAD PCM基本クロック(4/8MHz)の1/512，1/768，1/1024が使われます。基本クロックの設定はFM音源用のLSIであるYM2151のレジスタ$1Bに行い，クロックをいくつで分周するかはPPI(8255)のポートCに設定します。またパンポットもPPIポートCに設定します。

概要を理解したところでリスト1を見てください。これはAD PCM方式で格納されたファイルをAD PCMへ出力するプログラムです。入力するときは119行に再生するAD PCMファイルをフルパスで指定してください（DMA転送は1ブロックの最大長が65535バイトなので注意)。

ではプログラムの説明です。38～55行は指定のAD PCMファイルをバッファに読み込む処理です。60～66行でAD PCM基本クロックの設定をしています。OPMレジスタにデータを書き込む場合，まずデータを設定するレジスタ番号をOPM-REG-NO($E9A001)に設定し，OPMDATA($E9A003)にデータを書き込みます。リストにあるようにレジスタ番号$1Bの第7ビットが0で8MHz，1で4MHzになります。

89～92行がAD PCMの再生レートとパンポットの設定です。PPIポートCに設定するAD PCM関係のビット内容をリスト中に埋め込んでおきました。基本クロックと分周の組み合わせによる再生レートは，

	4MHz	8MHz
1/512	7.8K	15.6K
1/768	5.2K	10.4K
1/1024	3.9K	7.8K

となります。基本クロック8MHzを1/1024したものと4MHzを1/512したものは再生レートが同じです。95～105行がDMACチャンネル3の設定です。108～110行でDMACが転送動作を終了するまで待ちます。

このプログラムで問題となるのは65536バイト以上のデータ転送に対応していないことと，CPUがDMACの転送動作終了待ち

リスト2

```
 1:
 2:         moveq.l #0,d3                   * Work for BTC
 3:                                         * バッファ先頭アドレス
 4:         lea.l   adpcm_buffer(pc),a0
 5:         lea.l   aray_chane_tbl(pc),a1
 6: loop:
 7:         addq.w  #1,d3
 8:         sub.l   #65535,d2
 9:         ble     exit_loop
10:         move.l  a0,(a1)+                * 転送元アドレス
11:         move.w  #65535,(a1)+            * 転送サイズ
12:         adda.l  #65535,a0
13:         bra     loop
14: exit_loop:
15:         add.l   #65535,d2
16:         move.l  a0,(a1)+                * 転送元アドレス
17:         move.w  d2,(a1)                 * 転送サイズ
18:         move.w  d3,BTC3                 * 転送ブロック数
```

リスト1

```
  1: *
  2: * ADPCM出力サンプル
  3: *
  4:         .include        doscall.mac
  5:         .include        iocscall.mac
  6:
  7:         .text
  8:         .even
  9:
 10: OPM_REG_NO      equ     $E90001
 11: OPM_DATA        equ     $E90003
 12: PPI_PORT_C      equ     $E9A005
 13: ADPCM_COM       equ     $E92001
 14: ADPCM_DATA      equ     $E92003
 15: DMAC_CH3_BASE   equ     $E840C0
 16:
 17: CSR3            equ     $E840C0
 18: DCR3            equ     $E840C4
 19: OCR3            equ     $E840C5
 20: SCR3            equ     $E840C6
 21: CCR3            equ     $E840C7
 22: MTC3            equ     $E840CA
 23: MAR3            equ     $E840CC
 24: DAR3            equ     $E840D4
 25: BTC3            equ     $E840DA
 26: BAR3            equ     $E840DC
 27: MFC3            equ     $E840E9
 28: CPR3            equ     $E840ED
 29: DFC3            equ     $E840F1
 30:
 31: start:
 32:         clr.l   -(sp)
 33:         DOS     _SUPER                  * スーパーバイザ
 34:         move.l  d0,(sp)
 35:
 36: * ADPCM再生ファイル読み込み
 37:
 38:         clr.w   -(sp)
 39:         pea.l   pcm_fname
 40:         DOS     _OPEN
 41:         addq.l  #6,sp
 42:         move.l  d0,d1                   * ファイルハンドル退避
 43:         bmi     error                   * ファイルオープン失敗
 44:
 45:         move.l  #$ffffff,-(sp)
 46:         pea.l   adpcm_buffer
 47:         move.w  d1,-(sp)
 48:         DOS     _READ
 49:         lea.l   10(sp),sp
 50:         move.l  d0,d2                   * ファイルサイズ
 51:         bmi     error
 52:
 53:         move.w  d1,-(sp)
 54:         DOS     _CLOSE
 55:         addq.l  #2,sp
 56:
 57: * 基本クロックの設定
 58:
 59: wait_fm:
 60:         tst.b   OPM_DATA
 61:         bmi     wait_fm                 * 書き込み可能まで待つ
 62:         move.b  #$1b,OPM_REG_NO * 読み書きレジスタに$1Bを設定
 63: wait_fm2:
 64:         tst.b   OPM_DATA
 65:         bmi     wait_fm2                * 書き込み可能まで待つ
 66:         bclr.b  #7,OPM_DATA             * 最上位ビットクリア(8MHz)
 67: *       bset.b  #7,OPM_DATA             * 最上位ビットセット(4MHz)
 68:
 69: * 再生レート、パンポットの設定
 70: *
 71: * 8255(PPI)ポートC($E9A005)
 72: *
 73: *         7   6   5   4   3   2   1   0
 74: *       +---+---+---+---+---+---+---+---+
 75: *       |   |   |   |   | RATE  |  PAN  |
 76: *       +---+---+---+---+---+---+---+---+
 77: *                           --+--   --+--
 78: *                             |       |
 79: *                             |       00: 左右出力
 80: *                             |       01: 右出力
 81: *                             |       10: 左出力
 82: *                             |       11: 出力しない
 83: *                             |
 84: *                             00: 基本クロックの1/1024
 85: *                             01: 基本クロックの1/768
 86: *                             10: 基本クロックの1/512
 87: *                             11: 未使用
 88:
 89:         move.b  PPI_PORT_C,d0
 90:         andi.b  #%1111_0000,d0  * 下位4ビットクリア
 91:         ori.b   #%0000_1000,d0
 92:         move.b  d0,PPI_PORT_C   * 基本クロックの1/512
 93:                                 * パンポット=中央
 94:         move.b  #1,$e92001      * ADPCM再生停止
 95:         move.b  #$80,DCR3       * DMA転送に必要な設定
 96:         move.b  #$4,SCR3        *
 97:         move.b  #$05,MFC3       *
 98:         move.b  #$1,CPR3        * チャンネルプライオリティ2番目に高い優先度
 99:         move.b  #$05,DFC3       *
100:         move.b  #$72,OCR3       *
101:         move.b  #$ff,CSR3       *
102:         move.l  #adpcm_buffer,MAR3 * 転送元アドレス
103:         move.l  #ADPCM_DATA,DAR3   * 転送先アドレス
104:         move.w  d2,MTC3         * 転送サイズ
105:         move.b  #$80,CCR3       * DMA転送開始
106:         move.b  #2,$e92001      * ADPCM再生開始
107: wait_dma:
108:         move.b  CSR3,d0
109:         andi.b  #$90,d0
110:         beq     wait_dma
111:
112:         move.b  #$ff,CSR3
113: error:
114:         DOS     _SUPER          * ユーザーモード
115:         addq.l  #4,sp
116:
117:         DOS     _EXIT           * プログラム終了
118:
119:         .data
120: pcm_fname:
121:                                 * PCMファイル名(フルパス)
122:         dc.b    'a:¥adpcm¥jaja.PCM'
123:         dc.b    0
124:
125:         .even
126:
127:         .bss
128:
129: adpcm_buffer:
130:         ds.b    $ff00           * バッファサイズ $FF00
131:
132:         .end
```

をしていることでしょう。次に65536バイト以上のデータ転送に対応させてみることにします。複数のブロックを転送するにはアレイチェーン，リンクアレイチェーンを使うという話を前にしました。アレイチェーン，リンクアレイチェーンとも転送元アドレス，転送サイズをテーブルに記述します。両者の違いはテーブルの形式と転送終了条件です。リンクアレイチェーンはテーブルを不連続メモリ領域に取ることができます。サンプルではアレイチェーンを使ったプログラムを作成しました。

リストの変更点はDMACの設定に関する部分だけです。まず100行を，

```
    move.b  #$7a,OCR
```

とします。次に102行を，

```
    move.l  #aray-chanet-bl,MAR
```

さらに104行を削除して，そこにリスト2を挿入します。最後にバッファを増やしアレイチェーンテーブルを置く領域を新たに確保するので，128行のラベルadpcm-buffer以降の2行を削除して，

```
    aray-chanetbl:
        ds.b  6*3
    adpcm-buffer:
        ds.b  $ff00*3
```

を挿入してください。これでメモリが許すかぎりのAD PCMデータを再生できます。しかしいまのままではDMACの転送動作が完了するまでCPUは空ループを回しているので，AD PCMを再生しながらエディタを起動するといったことができません。

これを改良したい方のためにアドバイスします。X68000のDMACは転送動作が完了すると割り込みが発生します。そこでDMACチャンネル3の割り込みベクタ番号$6A(正常終了)，$6B(異常終了)のベクタエントリアドレスを変更しておき，その先でDMACの転送動作終了後の処理を書いておくようにします。その際，終了後の処理とAD PCMバッファは常駐終了させておかないとほかのプログラムを起動したとたんに破壊されてしまいます。注意してください。　　　　(影山裕昭)

Oh!X1991年6月号に掲載されていた「PC-9801のマウスをつなぐ」の記事を見て，PRO用マウスを改造してみようと思ったのですが，掲載されていた図と回路が異なっているため配線できません。配線方法を教えてください。　　　　**大阪府　橋本　智也**

質問にあるとおり，PRO用マウスにはいくつかのバージョンがあり，部品の配置が異なっているものがあるようです。しかし，それらの回路を検討すると回路自体に大きな変更はなく，部品のレイアウトを変更しているだけであることがわかりました。

これら以外にも違うマウスの回路パターンがあるかもしれません。そこで，どんな場合でも対応できるような対処手順を紹介しましょう。X68000マウスの基板を見て，基板に載っているコントローラが「MB88201H」という型番であることが確認できれば，これから述べる方法はどんな配置のマウス基板でも適用できます。

さてX68000のマウスは，マウス内部に内蔵されているコントローラ回路によってマウスの移動量およびスイッチ入力をシリアルデータに変換したあとにX68000本体に送信しています。一方，PC-9801シリーズのマウスは移動量データおよび，スイッチ入力をすべて別の信号線でパラレルに送信しています。PC-9801用のマウスによるパラレルデータはX68000のマウスにおいてコントローラICでシリアル変換する前のデータに対応していますので，PC-9801用のマウスをX68000に接続するにはマウス内部のコントローラICの対応する入力端子に直結すればよいことになります。

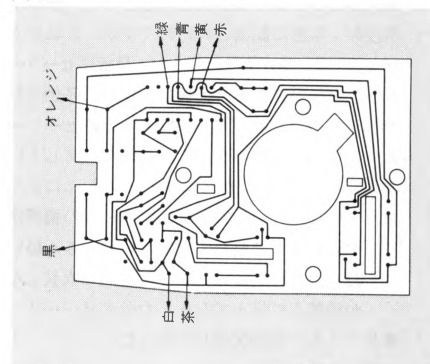
PC-9801用マウスの端子は9ピンDサブというコネクタが使われています。コネクタの端子に対応するコントローラICの端子番号を並べて示します。

端子番号	機能	コードの色	ICの端子
1	+5V	白	5
2	XA	青	12
3	XB	緑	11
4	YA	赤	14
5	YB	黄	13
6	LEFT	オレンジ	9
7	NC	—	
8	RIGHT	黒	1
9	GND	茶	4,8

ICの足の番号さえ見つければ，あとは上の対応表に従って配線していけばよいことになります。回路図は変わっていても，ICから出ている配線を追って対応する信号をつないでいけばいいのです。橋本さんのマウスでは図1のようになります。

ICの周りに配線が密集すると失敗しやすくなりますので，基板上のパターンがつながっている先を追いかけて，ゆとりのある箇所でハンダ付けするのがよいでしょう。たとえば，青(XA)，緑(XB)，赤(YA)，黄(YB)などのマウス移動量のデータ線は回転軸ユニットのほうから配線することができます。また，オレンジ(LEFT)と黒(RIGHT)のスイッチ入力のデータ線は押しボタンスイッチのほうから配線できます。さらに，白(+5V)と茶(GND)の電源ラインはマウスケーブルがハンダ付けされているあたりに黒色の電解コンデンサ(10μF, 16V)の両極の端子に配線するのがよいでしょう。それぞれ，上に挙げたICの各端子番号の足にパターンがつながっているかをよく確かめてください。

(三沢和彦)

図1

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を挙げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに解答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので電話番号も明記してくださいね。

宛先：〒108　東京都港区高輪2-19-13
NS高輪ビル
ソフトバンク株式会社出版部
Oh!X編集部「Oh!X質問箱」係

FROM READERS TO THE EDITOR

冬の楽しみはスキーやスケート，こたつにみかん。雪が降ったら雪合戦やつらら割り。霜柱をさくさく踏むのも嬉しいね。受験や卒業で忙しい人はあとひと息。体に気をつけてがんばって。春から始まる新しい生活のための準備ですからね。

◆Oh!Xを買い始めて1年たち，やっと「18日発売」ということを知りました。どうも売り切れが多いと思った。　坪根　聡(17)三重県

いままでは何日に買いに行っていたのでしょう？　これでもう買いそびれはありませんよね，ね。これからもよろしく。

◆遅ればせながら教育実習の報告です。私は小学校の1年生に配属されたのですが，とにかくセーラームーンが大流行でした。黒板にセーラームーンを描いてやると大喜び。結局，子供たちそれぞれの自由帳にのべ100体以上のセーラームーン(とその仲間)を描くはめになってしまいました(いっておくが私はまだせらむんにはハマっていない)。　上田　考一(22)福岡県

お疲れさまでした。そんなにたくさん描いたら，それぞれのキャラクターの衣装なんか細部まで覚えてしまったのでは……？

◆Ｙ「へえ，X68000持ってるんだ」
私「まあね(ふふん)」
Ｙ「じゃあ，セーラームーン見てる？」
私「……」
茂木　浩一郎(19)埼玉県

実はセーラームーン関係のおハガキが増え続けていて，担当者は嬉……い，いえ，なんでもありません……。でも，Oh!Xがアニメ雑誌になっちゃったらちょっと困るなぁ。

◆Oh!Xは僕の生きがいのようなものです。僕はいままでアニメばかり見ていて，女の子にもてませんでしたが，パソコンを買ってこの本を読むようになってから，女の子にモテモテで毎日がハッピーです。　川原　啓(18)群馬県

ほらほら，こういう人もいるじゃありませんか(でも，アニメが悪いわけじゃないと思うけど)。それはそうと，パソコンで女の子にもてる方法ってのを「詳しく」知りたいなぁ。

◆Oh!Xをマウスのマット代わりに使用するとマウスに色がつくのですが，どうしたらよいのですか？　天野　信幸(21)愛知県

回答：Oh!Xをマウスマットに使用してはいけません(いいんだけどさ……くすん)。

◆私は字がきたないのだが，このハガキを読む人はどのくらいの字まで解読可能なのでしょうか。ちなみに絵もきたない(個性的ともいう)。　谷口　浩史(19)北海道

ハガキはもちろん全部読んでます。だけどどこまで解読できるか「実験」なんてしちゃ，やだからね。

◆先日，私のX68000のハードディスクが認識されなくなりました。フロッピーで起動し，drive.xなどで調べてみても，ハードディスクは見あたらないのです。私はEXPERT-HDですから，外付けドライブのようなスイッチの入れ忘れはありえません。メモリスイッチにも異常ありません。で，どうなったかというと，カバーを外してホコリをはらったら，なぜか正常に動きだしました。いったい何なのでしょう。……実は前にも一度，これとまったく同じことが起きていて，同じ方法で元に戻っているのです。もうこんなことは起きてほしくないものです(一度でも起きてほしくなかったのですが)。　三浦　英樹(21)埼玉県

◆また寒い季節がやってきました。私のACE-HDちゃんのハードディスクは例年のように起動するのにだだをこねるんだろ～なぁ。しくしく。今年はこいつのために暖房器具でも買ってやるかぁ。　藤原　常雅(22)神奈川県

誰かが「パソコンは手間と金のかかる愛人」っていってましたねえ。暑いとだだをこねるし，寒いとすねる，メモリが足りないと文句をいい，ホコリをかぶってるとふてくされる……でも可愛いヤツでしょ。

◆私は広告を見るのが好きなのですが，見ながらあれもよい，これもよいと悩むとき，これ「しあわせ」なのよね。　阿部　敏仁(32)千葉県

ものを買うのって，いろいろ考えて迷ったりしているときがいちばん楽しいのかも。

◆このあいだ，2台目のVTRを購入して使い始めたんですが，どうしても気に入りません。その理由を挙げると……。録画予約の内容確認や取り消しがリモコンでできないこと，リモコン操作のたびにBEEP音(？)がピーピーうるさいことなど。3年前から使っている別会社のVTRのほうが使い心地がいいので，結局，画質が劣るにもかかわらずこの古いVTRをメインで使うことにしてしまいました。いまとなっては，カタログのスペックにばかり気をとられて，操作性をよく確認しなかったことが悔やまれます。パソコンやソフトウェアを選ぶ際にもこれらのことをよく考えたいものですね。　渡辺　久孝(25)大阪府

実際に使ってみないとわからないようなことも，たくさんありますしね。

◆やっと家からX68000を持ってきた。通信をやっているのですが，プロトコルの都合上，Macintoshでダウンロードし，友人のPC-9801を通して5インチにコピーし，それをX68000で解凍して使うという面倒なことをしている。でも面白いからいいや。　小林　勝(24)奈良県

文明の利器もなかなかややこしいなぁ。

◆5年前，X68000を買った。ファンがうるさかったが，パソコンを買ったのが初めてだったので，こんなものだと思っていた。去年，友達がEXPERTを買った。ファンが静かだった。今年11月，X68000を分解してみた。ファンが焼けていた。原因はファンの異常でした。所有しているXIGのものと同じだったので取り換えた

◀岡村　直也　兵庫県
すっかり「有名人」になった岡村さんは，読者の方に声をかけられたそうです。先月号は「お休み」だったので，がっかりした人も多いのでは……。

◀岩瀬　貴代美　福岡県
「受験雑誌(？)Oh!Xでは受験生の方へのはげましのお便りを募集」したわけじゃないけど，ちゃんと送ってくれました。みなさん，がんばってね。

ら静かになった。いままでの5年間はいったい……。　熊下　泰章(17)岩手県

まあ，過去のことは忘れて，忘れて。これからは静かで快適じゃありませんか。でも，確かにほかを知らなければ「こんなものだ」って思っちゃいますよね。もしかすると，こういってる私のパソコンも……。

◆やっとのことでXVIを手に入れることができた。理由はシルビアのローンがもう少しで終わるから。もう絶対こんな無茶なローンは組まんぞ，100,000円×24回なんて……。

南條　寿光(23)岐阜県

100,000円！　思わずゼロを数え直しちゃいました(カンマが打ってあるのに……)。ど，どうやって返済したの？　もしかして，すごい高給取りなのかな。ローンは便利ですけどあまり無茶はしないほうが……。

◆わがファミリーは皆，一括で，いきなり誰にも相談なしでものを買うというとんでもない慣習がある。古くは私がXICを持って帰ってきた。ビデオが壊れたので電気屋に修理に出したその日にほかのビデオを買って帰ってきた。母は「ちょっと買い物」といって，電気屋のトラックとともに帰ってきたときには冷蔵庫を買ってきた(ほかにエアコン，洗濯機もあった)。そして，すごいのは父。12月に入ってすぐ，誰かが来て，父は書類にサインしている。おもむろにどこに隠してたのか札束を渡した。「何事？」と思っていると，父が外に出ろという。そうです。車(スカイライン)を買っていたのです。はっきりいってこの快挙には一同，驚き。ちなみに我が家はローン禁止，現金一括払い，カードは誰も持ってないというすばらしい家です。

小宮　崇(21)埼玉県

現金一括払いで車購入！　車1台分の札束ってどのくらいの量なのでしょうか。そもそも「札束」なんか見たことないので，よくわからないのですが……。

◆雪国の利点──冬になると路面がアイスバーンになるので，暴走族がめっきり減る。これって，すごいことだと思いませんか？　有無をいわさぬ暴走禁止。対症療法しかできない警察よりよっぽどパワフル。　西崎　貴博(18)北海道

そりゃあ誰しも命は惜しいですからね。警察はまさか命までとるわけにはいかないだろうし。暴走族のいない，静かな雪国の冬ですか……いいなあ。

◆先日(といっても数カ月前)，80Mバイトのハードディスクを横浜まで買いにいきました。最近のハードディスクって安いですねー。80Mバイトで6万円を切るんですから。さて，買い物をすませ車へと戻……ない。……しぇー！　レッカーされとる～！　……最近のハードディスクは高い……ううっ。　井上　崇(21)神奈川県

路上駐車はだめだよん。

◆センター試験の日は大学が休みなので泊まりがけでスキーです。　矢野　啓介(19)北海道

そんなこと受験生に聞かれたら石が飛んでくるかも。受験生の皆さんは来年はこんなこと言えるように，がんばってね。

▲秋野　潤　静岡県
それがなんなんだ，っていわれたら困るけど「なんだかしあわせ」なのよね，うん。このキモチはユーザーじゃなきゃわからないのかもしれない。

▲佐田　匠　千葉県
「こういうキャラのほうが掲載されやすい？」って，うっ，スルドイ……いやいや嘘ですが。この本が出る頃はソフトも発売されてるといいですね。

◆12月号の愛知県の横田さん，「チャーハン」「焼飯」「ピラフ」の違いについて，私の知っている限りで答えましょう。まず，チャーハンと焼飯は同じものだと思ってよい，まあ中華風なのがチャーハンで日本風が焼飯だと思います。しかし，ピラフはまったく別物です。つまり，チャーハンが飯に具を入れて炒めたものなのに対して，ピラフは米に具を入れて炒めてから炊いたものです(と，思いますが)。

岡田　伸一(24)京都府

◆「ピラフ」はもともとペルシャ語で，中近東風肉飯だそうです。よって，チャーハンとピラフでは味付けが違うのです。

渡辺　幹司(20)三重県

◆レンゲで食べるのがチャーハン，割り箸で食べるのが焼飯，スプーンで食べるのがピラフ，ってことでどーだ！　よーするに中身はおんなじ，食べる人が決めるんです(24年間ずっと信じてます)。　佐藤　仁(24)静岡県

◆チャーハンと焼飯は中国語と日本語の差だけです。ピラフは油ではなくバターで炒めます。油とバターを混ぜる人もいますけど。また，矢野さんのいっていた「おすまん」は私のX68000では変換されませんでした。バージョン違いでしょうか。しかし「SHARP」と打つと「シャープ株式会社」にもなります。ライバル会社である「ソニー」や「東芝」まで一発で変換できます。あと，郵便番号の上3ケタで，そこの地名まで出ます。　亀田　徳隆(17)香川県

「違いのわからない」担当者の代わりに，たくさんの「お答え」をいただきました。だいたいが，ここで紹介したハガキの内容に集約されるようです。なるほどなるほど。それにしても，いろんなものが食べられる日本に住んでる私たちって幸せですよね。

◆12月号の「Oh!Xの読者の統計」を見ました。なんと驚いたことに島根県民のなかでOh!Xを読んでるのはたったの2人。とゆーことはオイラのほかにもう1人ってことですか？　なんてこったい。島根県民よ，もっとOh!Xを買ってくれい。それよりももっと驚いたのがCompactユーザーの数，たったの5人！　オーマイガッ！　となっちまうほど情けない台数なので，CompactのRAMははじめから8Mだとか，そんなオマケをしてCompactユーザーを増やしてください。頼みますよ，シャープさん。　北川　悟(16)島根県

こらこら，キミはアレをちゃんと読んだのかね？　あれはハガキをくださった読者のなかから任意抽出の500人のデータだから，その割合でいくと島根県の読者は……よくわからないけどざっと200人ぐらいはいるんじゃないかなぁ。とゆーことで，「少数派」の方々からの「嘆き(？)」のおハガキもたくさん来てしまいました。

◆いやぁ，本当に驚きました。斎藤学さんの訃報のことです。いつごろのことなのでしょうか。知らなかったのはボクだけか？　それにしても残念このうえないです。あの「闇の血族」の神秘的なMUSICには泣きました(感動して泣いたんです)。ご冥福をお祈りいたします。

松本　高佳(18)大阪府

もう斎藤さんの新しい曲を聴くことはできなくなってしまいました。彼の遺してくれたものを大切に，何かに役立てることができれば(形にするということでなくても)，彼も喜んでくださるかもしれませんね。

◆12月号の表紙のクルクル目のフランケンな犬のようなものについて……

わかっていること

・仲間がいるらしい(4月号より)
・暗いところであの目が光るらしい
・あいた口がふさがらないようだ
・職業はバーテンかもしれない

知りたいこと

・彼(彼女？)の名前
・どこに行けば会えるのか
・頭の角のようなものの果たす役割
・あいた口のふさがらない理由

西本　貴志(20)兵庫県

◆12月号の表紙の犬の名前は「フラン犬」とでもいうのだろうか。　中井　康雄(22)奈良県

◆あんな犬(かな？)に「いらっしゃい」とかいわれると困ってしまう。小海　昌伸(18)新潟県

「フラン犬」(仮称)へのおハガキは，ほかにも数通。次の登場が待たれますね。

◆土木工学科では，測量の時間に写真測量といって写真２枚を用いた立体視で行う測量を習う。そこで例のランダムドットを持っていき，友人にやらせてみたが，できる人はあまり多くはなかった。自分はといえば，ディスプレイ上でもできるまで成長してしまった。

小海　崇史(22)千葉県

もしかして，裸眼立体視ができないと単位がもらえない，とか……？

◆裸眼立体視って面白いですね。ところで僕には２通りの見え方がするのですが，僕の目は異常なのでしょうか？　中川　圭(18)千葉県

象さんがキリンさんになったりするのですか？　だとしたら，アナタの目は異常な目かもしれません。

◆僕の場合，裸眼立体視をするときは，まず「寄り目」にします。すると焦点がぼけるので，今度はそれを合わせるように，眼球を動かさないようにしていきます。あー，文章じゃ伝えにくい。　遠藤　勝博(22)宮城県

◆ランダムドットのジグソーパズルがあるそうですね。気分が悪くなりそうだ。

鈴木　恒一(22)茨城県

それって，すご～く難しいのでは？　それぞれのピースの絵を見ても，どのへんの部分なのかわからないと思うのですが……(想像しただけでキモチ悪くなりそう)。

◆このあいだ，MC68000を見ようと愛機をストリップにしたら，戻したあとになぜかネジが２本余った。おまけに日立製のチップだったなんて……。ごめんよ～。　金子　卓司(19)新潟県

ネジ２本分もダイエットさせられちゃったX68000。その後は元気ですか？

◆どうも１枚１枚ハガキに目を通しているのは本当のようだ(12月号のアフターレビューになぜか名前が載ってたんだもん)。

信垣　直嗣(19)大阪府

ひ，ひどい。信じてなかったのねっ！　ちゃんと全部(本当です)読んでますから，どんどんハガキ出してくださいね。

◆STUDIO Xなどで，いろいろあやしいことを書いて全国的に名前が知られたようなので夜も眠れず困っています。どうしたらよいのでしょうか。

1)　このままつっきる
2)　名前にうにうにと書く
3)　僕はあやしい人ではないと宣伝する

大島　大介(16)北海道

2)だとSTUDIO Xに掲載できないし(ペンネームは不可ですよ)，3)はきっと「無駄な努力」でしょうから，やっぱり1)しかないようですね(断言)。

◆「ていとうていとう」とは，『十訓抄』という説話に出てくる単語で，鼓の鳴る音を表す擬音なのだそうだ。日本の擬音もなかなか奥が深いと思った。　河村　憲昭(18)愛知県

擬音といえば，とっても不思議に思っているのが英語の鶏の声です。本当に「コッカードゥードゥルドゥー(これは昔，子供用の本に書いてあった)」って鳴くんでしょうか，英語を話す(？)ニワトリって……。

◆夜，布団に入ってビールを飲んでいたら知らないうちに寝てしまったんです。朝，なんとなくビール臭いと思って起きたら案の定，あたり一面黄色くなっていました。Oh!Xもそのビールの攻撃を受けベロベロになってしまい，しょうがないのでもう１冊買うことにしました。(もう１枚に続く)　宇野　高彦(26)神奈川県

◆(１枚目からの続き)このハガキがビールの攻撃を受けたOh!Xのです。というわけで，アンケートの内容は１枚目とほとんど同じです。

宇野　高彦(26)神奈川県

ということで，推薦ソフトとCGのアンケートには違うものを書いてくださいました。２枚目のほうもきれいなハガキだったので，そこは無事だったんですね。ちなみに，ほかにも２枚同じハガキをくださる方々がいるのですが，みなさん「寝ビール」とかで事故にあってるのでしょうか……？

◆あ。ついに柴田さんが妙ちけれんなことを始めましたね。見込んだとおりだ(笑)。個人的にSIONIIIと同じくらい期待するからね。

田中　幸雄(23)岡山県

◆柴田さんの普通とは違うものの見方にはいつも感動させられます。まさに「目のつけどころがシバタでしょ」という感じですね。

木下　孝雄(21)東京都

◆柴田さん，それ(12月号102ページ)ってば，ドライ・アイぢゃないですか……。

高橋　毅(21)埼玉県

ドライ・アイ(かも)の柴田さんは，いつも私たちに新鮮な感動を伝えてくれます。その感動をもとに自分でも何かスゴイことができればなあ，と思うのですが……。皆さんも柴田さんに負けずに，がんばってみてくださいね。そうしたらきっと，今度は柴田さんがそれに応えてくれると思いますよ。

◆Oh!Xに記事を書いている人たちって，何か謎めいてる人が多い感じがするのは私だけでしょうか？　(そーいうところで働けると楽しいでしょーね！)　畑山　保(20)千葉県

「神秘的」「ミステリアス」きゃあ，かっこいい。「変な人」「不気味」うっ……。いやいや，ごくふつーのヒトビトですよ，一部(全部？)の例外を除いては……。

◆毒を盛られて嘔吐する。そんな夢を見た。まったく身に覚えがないわけでもないので，気をつけようと思う。起きてみたら枕がゲロにまみれていた，とかいうことはなかったのが救いである。　中村　健(22)埼玉県

身に覚えがある？　あぶないあぶない。今日からココロを入れかえて清く正しく生きるってのは？　え，もう手遅れ？　う～ん。

◆よい子の私はいいつけどおり，ちゃんとSC-55を買ってきました(取り寄せ中なので，まだないけど)。もともと，X68000に白いMIDIはつけたくなかったので，SC-55を買おうかなと思っていましたけど……。あとは「人間マニュアル」の調達ですね……。これは難しそうだな。パソコンショップや本屋さんには売ってないだろうし……。こうなったら編集部に見合い写真でも送って「拾ってください」をするしかないなあ。いまならお値打ち価格０円のうえ，XVI，SC-55をセットでもれなくおつけします，とかで。

碓井　理恵(25)和歌山県

Oh!X編集部をはじめとして，当社には独身男性を各種取り揃えておりますが，内容不問，返品不可でもよろしいでしょうか？

◆実在弟がOh!Xの読者のようなので，何かこのコーナーに返事がほしいです。

坂下　実(23)神奈川県

それは「モーニング」誌に連載のマンガに出てくる実在弟(おとと)さんのことですね。残念ながら作者の方はX68000をご存じないようで，描かれていたOh!Xの裏表紙は違うパソコンでしたが……。いつか，Oh!Xだけじゃなくて，X68000も登場するといいなあ。

◆突然ですが，佐々木淳子の「青い竜の谷」(あすかコミックス)というマンガにX68000XVIが出ていました。舞台は1999年。８年後にXVIが現役で使われている……すごい(笑)。しかし，少女マンガに登場するパソコンといえばMacintoshだったんだけど(「姫100%」「チャイルドドライク・ワンダー」など)，ついにX68000も少女マン

◀小川　伸輔　宮城県
魅力的な，X68000SUPERちゃんの誘惑(?)にも負けずに勉強してるキミはえらい！　試験まであと少し，最後の追い込みまでがんばるのじゃぞ。

◀橋本　和典　東京都
彼女が持つのは「硬円盤内安全祈　「虫退散」のお守りです。プログラム開発中の人はコピーしてパソコンに貼るといいかも(もちろん彼女ごと)。

ガで認められるようになったか……(涙)。それにしてもこのマンガは面白いので，みんな読みましょう。　笹井　進也(22)神奈川県

マンガの世界にもX68000が浸透中？

◆社会人になって初めてわかる「勤労感謝の日」のありがたさよ。うるうる。

折田　正栄(24)大阪府

「勤労感謝の日」「文化の日」「体育の日」「天皇誕生日」……ありがたい日はたくさんありますが，そういうときに仕事をしてたりするとくやしさもひとしお……。ああ。

◆だ，誰か俺の代わりに受験勉強してくれ……。

佐々木　淳一(18)北海道

◆うちの学校は大学附属なので受験はしなくていいのですが，卒論を書かなければならない。冬休みはバイトしたいのに。誰か代わりに書いてください。　川本　健太郎(17)埼玉県

代わりにやってあげてもいいけど，結果は保証しませんよ。それでもいい？

◆秋葉原の中央通り，三菱銀行の前あたりで客引き(?)をしている名物おじさんをご存じでしょうか。体を前方に突き出し，ビラをぶん回しながら独特のボイスで「いかがっすかあ〜」。私の友人にも「あれを聞かないとアキバに来た気がしない」というのが多いので，今度ファンクラブを作ろうと思います。

清水　英明(21)神奈川県

あのおじさんの隠れファンは結構多いようですね。お願いしてサインをもらった人もいるとか……。でもみんなで押しかけてお仕事の邪魔ばっかりしちゃだめですよ。

◆私がピアノで遊び始めると，それまでベッドでまるくなっていたうちの猫はムクリと立ち上がり，なにやら迷惑そうにのそのそと部屋から出ていってしまう。まったく失礼なヤツだ。

中島　民哉(22)埼玉県

ピアニストの中村紘子さんちの猫なんか，跳びあがって逃げていってしまうそうです。猫ってピアノが嫌いなのでしょうか。

◆とうとう一児の父となってしまいました。なんかあっという間というか，簡単に子供ができて嬉しいかぎりです。ところで子供みたいに簡単にメモリも増設できないものでしょうかね〜。

金見　春彦(23)東京都

え，子供の増設ってカンタンなんですか？

◆えーと，6月19日に，広島の某ホテルで一生に一度の大宴会をT.Sさんとふたりで行うことになりました。これでやっと独り者の生活から抜け出せる。でもこれで，金が自由に使えなくなってしまう。はたしてNewX68000は買えるのであろうか。　松浦　隆明(28)広島県

おめでとうございます。ところで松浦さん，ハガキの表と裏ではあなたの年齢は3歳も違ってるんですが。きっとしあわせのあまり，やや錯乱なさっているに違いない，ということで特別に許してあげましょう。

▲占部　哲彦　広島県
「移植リクエストシリーズ」です。希望No.1は「ストII」とのことですが，占部さんが描くバルログとかザンギエフとかE.ホンダとかって…？

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できないこともあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

仲間

★「X68kマスターズ」では，新規会員を募集しています。活動の内容は，月1回ペースで発行されているディスクマガジン「X'm」(現在vol.19)を中心にしています。内容は会員間の意見の交換やプログラム，音楽を満載したものとなっています。入会したいと思われた方は，120円切手2枚を同封のうえ，下記の住所まで連絡してください。折り返し，入会案内の入ったサンプルディスクを送付させてもらいます。〒629-23　京都府与謝郡野田川町石川4452　大石方　X68kマスターズ「入会案内」係

★発足2年を迎えた，X68000ユーザーによるサークル「兎団」では，新規会員を募集中です。活動として，最新のPDS，情報，会員の投稿などを掲載したディスクマガジンの発行，いままでユーザーが築き上げてきた，膨大なPDSの無料コピーサービスがあります。また，オリジナルワープロなどの各種ソフトの開発も行っています。興味をもたれた方，いまならサンプルディスクマガジンとして2周年記念特大号（3枚組）を無料で配布しています。数に限りがありますので，なるべく早く官製ハガキでご連絡ください。なお，発送まで2週間程度時間がかかりますのでご了承ください。〒503-21　岐阜県不破郡垂井町宮代2840-1　田川　和義(18)

売ります

★シャープ製モデム「CZ-8TM2」を25,000円前後で売ります。新品同様，箱，付属品，すべてあります。希望価格を書いて往復ハガキで連絡してください。〒675　兵庫県加古川市神野町石守792-2　厚海　忍(19)

★X1用カラーイメージボード「CZ-8BV2」を送料込み16,000円で売ります（箱，付属品あり）。また，X1/X68000用熱転写プリンタ「CZ-8PC3」を送料込み21,000円（ケーブル，取扱説明書，モノクロリボンあり，箱なし）で売ります。まずは，往復ハガキで連絡してください。〒236　神奈川県横浜市金沢区釜利谷町1972-13　野崎　牧人(21)

★ドットプリンタ「CZ-8PK6」を25,000円，熱転写プリンタ「MZ-1P17」を5,000円で売ります（送料別，着払いで送付します）。連絡は往復ハガキでお願いします。〒440　愛知県豊橋市新吉町30　竹内　浩一

買います

★X1用RS-232Cボード「CZ-8RS」を10,000円で買います。連絡は往復ハガキでお願いします。〒329-44　栃木県下都賀郡大平町富田314-1　フラット大平寮　大島　靖浩(30)

★X1turbo用Z-BASIC＋64Kバイトバンクメモリ「CZ-141SF」を15,000円で買います。なお，64Kバイトバンクメモリのみの場合は，10,000円で買います。連絡は往復ハガキでお願いします。〒272　千葉県市川市国府台4-7-29　水野　一雄(23)

★X1用FM音源ボード「CZ-8BS1」を送料込み12,000円前後で買います。箱はなくてかまいませんが，付属品はつけてください。連絡は希望価格を明記のうえ，往復ハガキでお願いします。〒631　奈良県奈良市富雄北3-20-16　辻村　秀臣(19)

バックナンバー

★Oh!X1990年8,9月号を送料込み各1,500円で買います。なるべく両方を売ってくださる方を希望しますが，バラでもかまいません。連絡は官製ハガキでお願いします。〒173　東京都板橋区仲町41-4　#203　梅谷　信彦(22)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々の意見を紹介しています。今月は12月号の内容に関するレポートです。

●特別企画にあった「Oh!XとOh!Xの読者の統計」を見ていて思ったのが，案外「意外性にとんでいない」ということでした。Oh!Xを読んでいるとだいたい予想がつくというか，なんというか。でも，いろいろな意味で幅広い読者層がいるということは，Oh!Xが冒険していくうえでなかなか心強いことかもしれませんね。現状のまま留まらず，これからもひたすらOh!Xらしく，さまざまなことにチャレンジしていってくださいね。がんばってついていきます，はい。あと，新製品紹介にあった「Y300-A」について，私がこうしたDTPソフトに期待したいのは，ちょっとした文章を作るときにはワープロ代わりにもなり，必要とあれば本格的なものも作れる，つまり1本のソフトで文章を書き，印刷するという仕事であればあらゆる場合に応じて対応できてほしい，ということです。こうした試みは，プロの世界でしかできなかった本格的な印刷物の作成を素人でも可能にする一歩だと思います。処理速度の問題や表示方法など，ハードウェア的にも克服しなければいけない問題も多々あると思いますが，メーカーさんにはがんばってほしいです（もちろん価格もね）。

前田　秀樹(19)　X68000 XVI，PRO，MSX，MSX2　京都府

●12月号の特別企画「ショートプロ大集合」はよかったです。手軽に楽しめるうえ，いつのまにかプログラミングのコツまで身につくような気がしました。今後も年1回ぐらいは，このような企画を読者も参加させる形でやると面白いと思います。しかし，現在ではこのような記事が少なくなりましたね。ゲーム紹介やハード，ソフトの活用，あるいは言語の説明記事が，やたら目立つパソコン雑誌ばかりになってきたような気がしてなりません。プログラミングといえば，5年前，MSX2でRPGを作ったことがありました。これは自分でも気に入っていて，最近になっても遊ぶことがあります。VRAM書き換えによる文字変形，着色，ファイル操作，BIOSコールなど，文字どおり当時の技術の結集，いや，私はあのゲームを作るために技術を身につけていったのかもしれません。誰にでもこのような思い出のあるプログラムはあるのでしょう。プログラミングで悩んだとき，そのプログラムを思い出してはエネルギーを得る，そんなプログラムを作れれば幸せですよね。

宍戸　輝光(19)　X68000 PRO，MSX2 東京都

●特別企画にあった「裸眼立体視（ランダムドット）」は面白い！　しかし，市販されているものは，どうしてあんなに高価なんでしょうかね。こんなに簡単に作れるのに。アルゴリズムについては，説明を読んでもいまいち理解できずにいますが。X68000にはプリンタがつながっていないため，現在，ハイパーカードに移植して遊んでいます。

中島　奨(26)　X68000 PRO II，Macintosh SE/30

●12月号の特別企画についてですが，掲載されたプログラムが悪いとはいいません（むしろよいものが多かったと思います）。しかし，もう少し，小粒なツールというか，あったら便利なプログラム，といったものを掲載したらよかったのではないでしょうか。ある種のフィルタやツールなど，「山椒は小粒でもピリリと辛い」的なものは，本当に役立つのですから。

高橋　毅(21)　X68000 PRO，MSX2　埼玉県

●ショートプログラムって「必要に迫られて作るもの」と「なんとなく作るもの」の2種類あるんですね。12月号の特別企画を読んでそう思いました。前者は「STRFIND.C」や「パワーダウンマネージャ」であり，後者は「MAGICAL TRIANGLE」などが当てはまりそうです。両者の違いは「目的性の有無」であり，前者ならツール，後者ならゲーム関係が多いようです。全体的に見ていくと今回の特別企画では，後者のパターンが多かったですね。まあ，こういうなんとなく作ったものは，なかなか人前に出せないものですけど。

中村　健(22)　X68000 ACE-HD，AMIGA 500，PC-386GS，MSX2＋　埼玉県

●なにか精通している分野があると，ショートプログラムでも面白いものができるものですね。企画自体はよかったのですが，Oh!X 5周年ということでもっとぶっ飛んだ企画でもよかったのでは，とも思いました。あと，microOdysseyの言葉についての話は，共感する部分が多くて楽しく読むことができました。私自身も，本などで使われている言葉遣いにはうるさいほうなので，（ふ）さんのおっしゃることにいちいちうなずきながら読んでしまいました。

矢野　啓介(19)　X68000 XVI，MZ-2500　北海道

●12月号の「猫とコンピュータ」は，とても興味深かったです。猫に肩があるのか，あるなら肩こりをするのか，などというふだんあまり考えたことがないようなテーマで，面白かったですね。こういった人々の盲点ともいえる疑問は，世の中に満ちあふれているのでしょう。なにげなく歩いている街中にも不思議が渦巻いているのかもしれませんね。

志田　健(17)　X68000 SUPER　東京都

●12月号で印象に残ったのは，X-OVER NIGHT「不良資産」でした。私は，一度手に入れたものを処分できない性格で，「なにか役に立つだろう」とか「これはそのうち値打ちが上がるだろう」と考え，いわゆるゴミのたぐいしか捨てません。私は，ものを買うときに結構慎重に選ぶのですが，ものを捨てるときも慎重であるのは，やはりよくないことだと反省しています。私の机の棚を見ると，小中学校の頃からずっとそこにあるものが多いのです。今年社会人になったのに，まったく成長していないようで恥ずかしかったですね。今年は本当の大掃除ができそうです。

村上　晃(23)　X68000 XVI　岡山県

ごめんなさいのコーナー

1月号　Oh!X LIVE in '93

P.67　リスト1の「セーラームーン・ムーンライト伝説」が，ZPP.Xで展開したZMSファイルで掲載されていました。ですから，リスト1はZPP.Xで展開する必要がなく，そのまま演奏させることができます。

バグに関するお問い合わせは

☎03(5488)1311(直通)

月〜金曜日16：00〜18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

ディプレイにある広大な空間をどう利用するか

▼現実にあるものを模倣する。模倣をするだけでも，その苦労は途方もないものがあります。しかし，実現不可能ではありません。丁寧に物理現象を追いかければ再現可能だし，それを表現できる能力がコンピュータにあります。しょせんは仮想世界さ，とうそぶいてもリアルな画像の説得力は，皆さんご承知のとおりでしょう。

難しいからね，とあきらめていては進歩がありません。なにができて，なにができないか見極める意味でも，今月の特集をじっくり読んでみてください。きっと新しい発見があるはずです。そして，発見したらとにかく実践あるのみ。完成したものは，目指すものよりもはるか遠いところにある，ぎこちないものかもしれません。しかし，現在はぎこちなくても，いくらでも近づいていく可能性はあります。皆さんの力で，その可能性を切り開いていこうではありませんか。

▼さて，仮想だ，現実だ，などという堅苦しい議論はさておき，いま目の前にある現実として1992年度GAME OF THE YEARの投票があります(すごい強引)。ずらり並んだノミネート作品，どの作品に投票するかはあなたの自由です。自分の思い入れのある作品に熱い1票をぶつけましょう。去年とは応募方法が多少異なっているので，応募要項をよく読んで間違えのないようにしてくださいね。

従来どおりの勝手にGAME OF THE YEARとは別の，ゲーム回顧録の代わりに設けられた「読者レビュー」の投稿も忘れてはいけませんよ。このゲームはこんなところが面白い，このゲームのここには注意しておこう，といった読者の皆さんが実際に遊んでみて感じたポイントがあるはずです。また，誌面に登場して目立てる機会でもあるので，がんがん投稿してくださいね。GAME OF THE YEARの締め切りは2月18日必着ですから，その点もお忘れなく。

▼「大人のためのX68000」は著者多忙のためお休みさせていただきました。

投稿応募要領

- 原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。
- 投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒108 東京都港区高輪2-19-13 NS高輪ビル

ソフトバンク出版部

Oh!X「㋢㋷㋮㋱」係

SHIFT・BREAK

▶プレゼントにもありますが，今年もなんとか上昇気流を出すことができました。ぱちぱち。毎年いろんなものを削って作っているわけですが，できてしまえばそんな苦労も忘れ，またやりたくなってしまう(笑)。しかし人間は思いっきり恥をかき倒さな成長せんばいね，と切に感じる今日この頃です。やるだけやらな先に進めんとって。ほんと。　(哲)

▶編集室にはX/FM兼用の冷蔵庫がある。ジュースとお菓子専用だ。で，先日中身の整理をした。1990年3月製造のオレンジジュース。賞味期限を1年過ぎたゼリー。「信じらんね〜」といいながら捨てた。だが数時間後には謎の腹痛に苦しめられる私の姿が。関係ないけど，古いオレンジジュースってマーマレードの匂いがするんだぜ。関係ないけど。　(浦)

▶ダグラス・アダムスのSFコメディ小説「銀河ヒッチハイクガイド」を何年ぶりかに読み返した。面白い。変。感動。大爆笑。続編の「宇宙の果てのレストラン」「宇宙クリケット大戦争」は持っていない。出版社に問い合わせたら絶版だって。読者の皆さまーっ。東京か埼玉で売ってたらハガキに書いて，その書店の場所を教えてちょうだい。　(善)

▶X68000に毛布をかぶせて，足を入れると暖かいのは事実ですが，ユタンポ代わりに使うには向かないようです。私はそれでX68000の側壁を破壊してしまいましたから。けっとばしちゃったのかな？ところで，某インテル系の最新型CPUでは，稼働中の表面温度は100度近くなることもあるんだそうですね。うーむ，湯沸かし機代わりに1台ほしい。(で)

▶最近リムーバブルハードディスクの寝起きが悪い。早くも故障？　これだから外国製品は，と思っていたが，ふと思いついてドライヤーの温風をスロットに吹き込んでみると，あっさり動き出した。買ったのは夏だったしねえ。さて，AMIGAとX68000×2で共用したいんだけど，SCSIセレクタって市販されてないみたい。確かに危ないけどさ。　(A.T.)

▶もう1993年なんですよねえ。なんかウラシマ効果にあったような気分。1992年はねえ「24人のビリーミリガン」は面白かったなあ。「浴室の窓から彼女は」もよかったし，「タタール人の砂漠」も私の趣味だし，「オモライくん」は復活したし，「コンピュータ社会と漢字」は参考になったけど，やはり忘れられないのは「朝のガスパール」だな。　(K)

▶先日の新聞を見て驚いた。今年あの宇宙戦艦ヤマトが復活するという。デスラーが主演の映画とかいままでにいろいろ噂があったが，今度は本当なんでしょうね，西崎さん。でも，アクエリアスに沈んだヤマトを引き揚げるなんて考えずに，まったく新しいヤマトを見せてほしかった。やっぱり粉々に砕かない限りヤマトのやすらぎはないのかなあ。(KO)

▶1日の区切りすらあいまいな忙しさの合間を縫って，バレエを観に行った。年末定番の「くるみ割り人形」。変化に富んだ演出の面白さもさることながら，鍛えられ緊張感を持った肉体の美しさには，なによりも激しい表現力がある。睡眠不足の頭にさえ呼びかけてくる響きがある。それにしても，バレリーナの足ってどうなってるのかな？　(ふ)

▶忙しいのに風邪をひいた。いちばん困ったことといえば，タバコが吸えなくなったことだ。僕の場合，風邪をひくと，まず喉にくるので休煙をよぎなくされる。病気のときぐらいあきらめろよ，といわれるかもしれないが，すでにニコチンとタールに侵された体にとっては非常につらい。やっぱ馬鹿は死ななきゃ直らないかなあ。　(編集部員唯一の喫煙者J)

▶この間は香川県。そして，今度は高知県。先輩と友人のめでたい出来事だから，うれしいことはうれしいんだけど，さすがに財布の中が気になりだしてしまう。東京と四国だと往復するだけでも相当だし，そのほかにもやっぱり雑費がいろいろとかかる。でも，やるほうはもっとお金がかかっているんだからなあ，結婚式や披露宴って。　(A)

▶凄いんだがどうも煮え切らなかったポピュラスII。それがChallenging gamesになって疑惑は氷解した。たとえばゲーム開始直後に四方から騎士が襲ってくる。やられるまえに周りに壁を築くのだ。地割れの上は壁ができないので花を使い……。技を駆使する楽しみがある。AMIGAのストIIはパターンは吸い出しても動きまでは無理だったみたい。　(U)

▶地元池袋のメトロポリタンプラザにJリーグオフィシャルショップがオープンした。ついつい余計な買い物（ミニチュアのサッカーボールとか）をしてしまい，出勤が遅れることも。以前，microOdysseyで日本のサッカーが弱い理由はうんたらかんたらと書いたことがあったが，とりあえずあれはなかったことにしたい。ガンバレニッポン！　(T)

microOdyssey

先月から始まった68020ボードの製作について少々補足しておこう。意外に批判的な声が少ないのだが，Oh!Xがこのような本体の改造にあたることを行おうとしていることに驚いた方もいるのではないだろうか。ことに，もうしばらく待てばそんな苦労をせずとも32ビットマシンが現れるのがわかっていながら，である。

今回の製作記事は実はもう1年前に企画されたものだ。内容にしても，X68000が発表された当時に関連技術の話題として何度か紹介したアクセラレータボードとほとんど同じものだ。6年前に提示したものをなぜいまさらという疑問も浮かんでくるだろう。

32ビット化という問題はX68000がそのCPUを決定した時点で宿命づけられた課題である。

雑誌を作っていくうえでも，いかにして将来的な互換性を確保するかというのもひとつのテーマだったといっていい。直接ハードウェアをいじっているようなものにはたとえ優秀なプログラムでも批判されたし，当時の編集長の指示で当分のあいだマシン語の入門は行わないことになっていた。扱うことがあってもできるだけIOCSを通すという方針だ。ソフトウェアについてもシステム周りには深入りしない。8ビット機ではOSまで作っていた連中が素直にメーカーの意向に従っていたのにはそういう経緯がある。そう，ある時点までは。

本来，家電製品というのは，中身を開けただけで保証がきかなくなるものだと思っていい。しかし「コンピュータというものはそういうものではない」というのも事実だ。RAMの増設などで本体を開けたことのある人も多いと思う。ドライブの設定を変えたり，メモリを増設をしたり，拡張スロットを付け加えたりと，ハードウェアの付加，置き換えでシステムを強化していくことは当たり前の世界だ。

そこで期待されるのがサードパーティのハード屋さんなのだが，V70ボードにしてもPOLYPHONにしても行儀のよいボードに収まっている。海外ではもっと怪しい代物が出回っている。その手のもので，しかもアメリカ産などとすると確実に本体故障のトラブルがいくつか発生しているはずなのだが，流れは止まることはない。

しかし，パソコンというものを考えると，そういった怪しい動きがあるほうがむしろ正常であるといえる。X68000などはまだ水面化で怪しい動きが見られるほうだが，表面だったところにはなかなか出てこない。X68000ユーザーの最大の源泉となっているのはほかならぬ，あの「セミ手作りマシンMZ-80K」からの流れなのである。これは異常である。

ちゃんと「もっと怪しい動きを作ること」というのも今回の68020ボードの目的のひとつだ。「仮にアクセラレータが完成しても32ビット機が発売されていればまったく需要のないものではないのか？」という当然の疑問を抱く人は正しい。ついでにもっと先を見てもらえるとさらにうれしい。

あのときから我々は5年後を夢見てそれに備えてきた。それももうすぐ終わる。そしてさらに次の段階に突入しようとしているわけだ。

個人的に今年の目標は「手加減はしない」「他人のことは考えない」に決めている。やらなければならないことは山ほどあるのだ。　(U)

1993年3月号2月18日(木)発売

特集　X-BASICを学ぶ
・X-BASICの基礎
・X-BASICによるX-BASIC外部関数作成

新製品紹介　MIDI音源モジュールSC-33/QY-20

MIRAGE System Model Stuff
全機種共通システム
シューティングゲームコアシステム作成法

バックナンバー常備店

東京	神保町	三省堂神田本店5F 03(3233)3312
	〃	書泉ブックマートB1 03(3294)0011
	〃	書泉グランデ5F 03(3295)0011
	秋葉原	T-ZONE 7Fブックゾーン 03(3257)2660
	八重洲	八重洲ブックセンター3F 03(3281)1811
	新宿	紀伊国屋書店本店 03(3354)0131
	高田馬場	未来堂書店 03(3209)0656
	渋谷	大盛堂書店 03(3463)0511
	池袋	旭屋書店池袋店 03(3986)0311
	八王子	くまざわ書店八王子本店 0426(25)1201
神奈川	横浜	有隣堂横浜駅西口店 045(311)6265
	〃	有隣堂ルミネ店 045(453)0811
	藤沢	有隣堂藤沢店 0466(26)1411
神奈川	厚木	有隣堂厚木店 0462(23)4111
	平塚	文教堂四の宮店 0463(54)2880
千葉	柏	新星堂カルチェ5 0471(64)8551
	船橋	リブロ船橋店 0474(25)0111
	〃	芳林堂書店津田沼店 0474(78)3737
	千葉	多田屋千葉セントラルプラザ店 0472(24)1333
埼玉	川越	黒田書店 0492(25)3138
	川口	岩渕書店 0482(52)2190
茨城	水戸	川又書店駅前店 0292(31)0102
大阪	北区	旭屋書店本店 06(313)1191
	都島区	駸々堂京橋店 06(353)2413
京都	中京区	オーム社書店 075(221)0280
愛知	名古屋	三省堂名古屋店 052(562)0077
	〃	パソコンΣ上前津店 052(251)8334
	刈谷	三洋堂書店刈谷店 0566(24)1134
長野	飯田	平安堂飯田店 0265(24)4545
北海道	室蘭	室蘭工業大学生協 0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある『新規』『継続』のいずれかに○をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください。なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700

Oh!X 2月号
■1993年2月1日発行　定価600円(本体583円)
■**発行人**　孫　正義
■**編集人**　橋本五郎
■**発売元**　ソフトバンク株式会社
■**出版事業部**　〒108 東京都港区高輪2-19-13　NS高輪ビル
Oh!X編集部　☎03(5488)1309
出版営業部　☎03(5488)1360 FAX 03(5488)1364
広告営業部　☎03(5488)1365
■**印　刷**　凸版印刷株式会社

満開の電子(でんこ)ちゃん

作・え　岡村(おかむら)祭(まつり)

村橋　裕幸（岐阜県）

２年前の７月に私の運命を決めるダイレクトメールが家に届いた。それは、電脳倶楽部の定期購読の案内であった。私は内心不安だったが、誘惑に負けて購読してみることにした。まもなくすると水色の封筒に入ったディスクが送られ、起動してみると、「ブラボー！」と叫ぶほど素晴らしかった。

今ではすっかり電脳倶楽部の虜になってしまい、学校の試験期間中でもディスクが送られてくると、電源オンですぐ起動してしまう体質になってしまった。

あなたも電脳倶楽部を購読し、満開製作所の野望に○○○○○○。

1月20日
第1回サポートサービス(無償)開始

日本語ワードプロセッサ

サンダーワード

ThunderWord ver 1.0

あなたはもう雷語(サンダーワード)の使い方を知っている!
かな漢字変換は標準FEPの**ASK68K**に準拠
ED.Xと**MicroEMACS**のコマンド体系
X68000ビットマップディスプレイ機能を活用
ルビ・アンダーライン機能
最大32ファイルを同時編集
最大15までの水平分割ウインドウ
フレンドリーな辞書登録機能
プリンタはCZ,ESC/P,NM,PC-PRに対応
縦・横印刷機能、印刷プレビュー機能

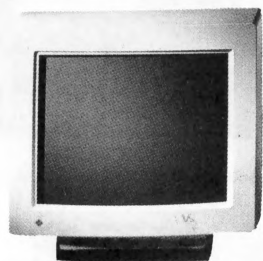
標準価格20,000円(税込)
(本体19,417円)

商品・通販のお問い合わせは
〒171 東京都豊島区長崎1-28-23 Muse西池袋2F TEL(03)3554-9282 FAX(03)3554-3856

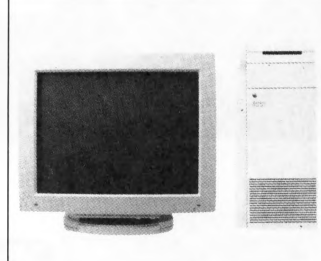
(株)満開製作所

宇都宮新世界発見ワークステーションショップ

BASIC HOUSE
KEISOKUGIKEN Corp.

全国に先駆けてカリフォルニア産の人気マシンを一同に展示中お誘い合わせの上ご来店ください

SUN SPARC Station

IRIS Indigo

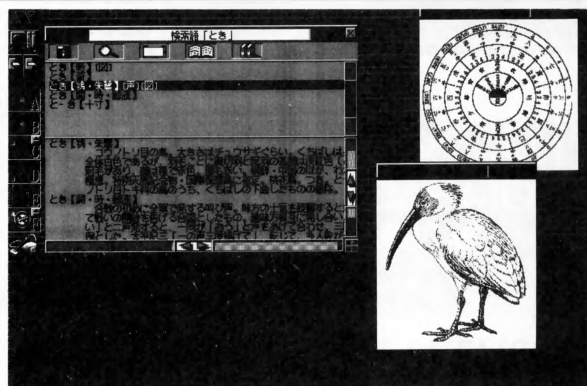
NeXT Station

Apple Macintosh

OPEN 12周年記念セール開催中 1月末日まで

X-68000用CD-ROM Drive

X-68000, Macintosh
ドライバーソフト
CDキャディ
SCSIケーブル

38%OFF
標準価格¥128,000

記念特価　超目玉商品　限定50本限り　特価¥79,800

X-68000 計測技研オリジナルセット 12周年記念特別価格　限定即納できます!!

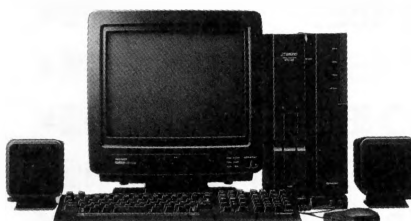
X68000CompactXVI/HD120

2.5"Quantum120MB内蔵

CZ-674C-H	298,000
Go Drive 120MB	198,000
定価合計	496,000

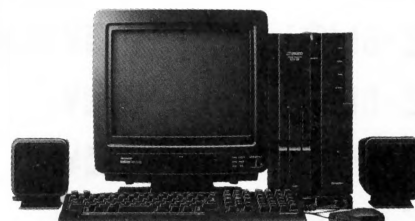
36%OFF　318,000

X68000XVI/HD240

3.5"Quantum240MB内蔵

CZ-634C-TN	368,000
Quantum 240MB	198,000
定価合計	496,000

36%OFF　368,000

X68000XVI/HD425

3.5"Quantum425MB内蔵

CZ-634C-TN	368,000
Quantum 425MB	298,000
定価合計	496,000

33%OFF　448,000

―― 好評発売中 ――

X68000 CD-ROM第一弾!!

FREE SOFTWARE SELECTION

中味は買ってからのお楽しみとにかくすごいものがたくさん入っています

超目玉 3台限り

CZ-8PC5-BK 熱転写カラー漢字プリンタ

定価96,800円　49,800円　50%OFF

X68000 PROSHOP

株式会社計測技研

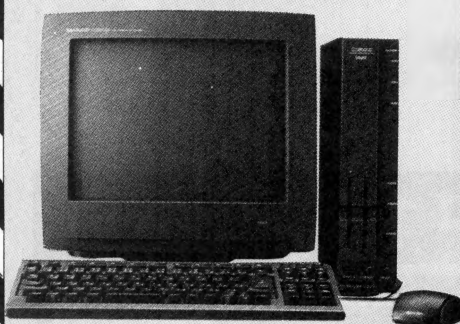
本社ショールーム　〒321 栃木県宇都宮市竹林町503-1
TEL0286-22-9811　FAX0286-25-3970

研究開発部門　First Class Technology
〒320 栃木県宇都宮市京町11-18
TEL0286-38-0301　FAX0286-38-0305

Sunnyvale営業所　875 Cumbertand Drive Sunnyvale, CA. 94087
TEL408-720-1573　FAX408-720-1576

マイコンショップ川口

☎048-225-1718

（消費税別）

New X68000
COMPACT XVI
~~¥298,000~~

CZ-674C-H	定価¥298,000
CZ-608D-H	定価¥ 94,800
AV-090-SC	定価¥168,000

超特価 TEL下さい

CZ-634C-TN	㊪368,000
CZ-644C-TN	㊪518,000

ソフト各種超特価ご奉仕中

CZ-219SS OS-9/X68000	定価¥29,800
CZ-213MS MUSIC PRO68K	定価¥18,800
CZ-214MS SOUND PRO68K	定価¥15,800
CZ-215MS Sampling PRO68K	定価¥17,800
CZ-220BS DATA PRO68K	定価¥58,000
CZ-224LS The 福袋 Ver2.0	定価¥ 9,980
CZ-225BS Multiword	定価¥32,000
CZ-251BS Hyper word	定価¥39,800

開店10周年記念

大奉仕キャンペーン実施中!!

品名	定価	売価
CZ-674C-H本体	¥298,000	大特価
CZ-634C-TN本体	¥368,000	大特価
CZ-644C-TN本体	¥518,000	大特価
CZ-608D-Hディスプレイ	¥ 94,800	大特価
CZ-606D-TNディスプレイ	¥ 79,800	大特価
CZ-607D-TNディスプレイ	¥ 99,800	大特価
CZ-614D-TNディスプレイ	¥135,000	大特価

プリンター

CZ-6VT1	特価¥ 47,700
CZ-8PG1	特価¥ 86,800
CZ-8PG2	特価¥106,900
CZ-8PK10	特価¥ 66,800
CZ-8NS1	特価¥141,000
CZ-6BC1	特価¥
CZ-6BG1	特価¥
CZ-6BP1	特価¥
CZ-6BP2	特価¥ 34,400

ラムボード

CZ-6BE2A	定価¥59,800	特価¥ 44,900
CZ-6BE2B	定価¥54,800	特価¥ 41,100
CZ-6BE2D	定価¥54,800	特価¥ 41,100
CZ-6BE1B	定価¥28,000	特価¥ 21,000
CZ-6BE2	定価¥79,800	特価¥
CZ-6BE4C	定価¥98,000	特価¥
PIO-6BE1-A	定価¥25,000	特価¥
PIO-6BE2-2M	定価¥50,000	特価¥
PIO-6BE4-4M	定価¥88,000	特価¥
SH-6BE1-1M	定価¥25,000	特価¥

ファイル

CZ-6MO1	定価¥450,000	特価¥
CZ-64H	定価¥120,000	特価¥
CZ-68H	定価¥160,000	特価¥

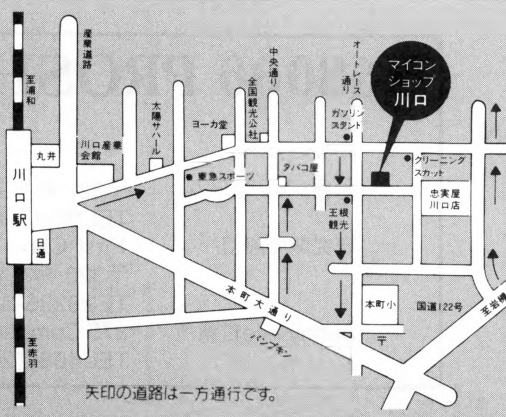
その他機種

CZ-8NS1 カラーイメージスキャナ	定価¥188,000	特価¥
JX-220X カラーイメージスキャナ	定価¥168,000	特価¥
CZ-6BN1 スキャナ用パラレルボード	定価¥ 29,800	特価¥
CZ-6VT1 カラーイメージユニット	定価¥ 69,800	特価¥
CZ-6BV1 ビデオボード	定価¥ 21,000	特価¥
CZ-8TM2 モデムユニット	定価¥ 49,800	特価¥
CZ-8NJ2 インテリジェントコントローラ	定価¥ 23,800	特価¥
CZ-8NM3 マウス・トラックボール	定価¥ 9,800	特価¥
CZ-8NT1 トラックボール	定価¥ 6,888	特価¥
CZ-8NJ1 ジョイカード	定価¥ 1,700	特価¥
CZ-6BC1 FAXボード	定価¥ 79,800	特価¥
CZ-6BM1A MIDIボード	定価¥ 26,800	特価¥
CZ-6BP1 数値演算プロセッサ	定価¥ 79,800	特価¥
CZ-6BP2 数値演算プロセッサ	定価¥ 45,800	特価¥
CZ-6TU-BK-GY RGBシステムチューナー	定価¥ 33,100	特価¥

★クレジット回数1〜60回まで設定自由

回数	1	3	6	12	15	20	24	36	42	48	54	60
金利(%)	2.5	2.9	3.9	5.4	8.4	10.9	11.4	15.9	19.9	20.9	25.9	26.9

中古品も取扱っております。

ショップ専用☎048-225-2500

通信販売をご利用の方

―全国通販―

通信販売をご利用の方は、売値の変動がありますので在庫、値段をあらかじめ確認のうえ電話で、商品名及びお客様の住所・氏名・電話番号をお知らせ下さい。

ACCESS for X68000

V70アクセラレータならすぐに展覧会が開けます。

V70アクセラレータが数値演算で高いパフォーマンスを誇るのは、クロック20MHzのV70CPUを搭載し、さらにAFPP(フローティング・ポイント・プロセッサ）を標準装備しているからです。特にコンピュータ・グラフィックスの世界では、その実力を十二分に発揮することができるでしょう。写真のグラフィックスでは、実行速度で約45倍のパフォーマンスを記録しました。開発環境に関しても、アセンブラ、リンカはもちろん、ソースコードデバッガやフロートエミュレータ・コマンドシェルと、V70の特徴である仮想記憶、メモリプロテクション等をサポートする充実した開発環境が整っています。V70アクセラレータは、一所懸命に作ったプログラムの実行結果をすぐに見たい！ というあなたの願いを、きっとかなえてくれるボードです。

```
for(x=0;x<512;x++){
        for(y=0;y<512;y++){
                X=0.0;
                Y=0.0;
                for(t=1;;t++){
                        if(t==512){
                                break;
                        }
                        Q=X*X-Y*Y+x*T+p[0];
                        R=2*X*Y+y*U+p[2];
                        if((Q*Q+R*R)>4.0){
                                break;
                        }
                        X=Q;
                        Y=R;
                }
                psetptr.x=x;
                psetptr.y=y;
                psetptr.color=((t)%256);
                PSET(&psetptr);
                palat[y]=(unsigned char)psetptr.color;
        }
}
```

上記グラフィックス（フラクタル）作成の為のプログラム（主要演算部分）

上記グラフィックスの描画速度比較

X68000(10MHz＋FPP無し)＋FLOAT2.X……約27時間10分

X68000(10MHz)＋VDTK-X68K……………約**37**分！

VDTK-X68Kの仕様

- V70 CPU(μPD70632)
  20MHz 32ビットマイクロプロセッサ
- V70 AFPP(μPD72691)
  フローティング・ポイント・プロセッサ
- メインメモリ(DRAM)2Mバイト
  同一ページ内のアクセスはNo Wait
- 共有メモリ(SRAM)128Kバイト
  X68000との通信用
- 併行動作 X68000とV70は、併行に動作することが可能。
  データの受け渡し処理のために双方向ハンドシェークI/Oポートを搭載。

同梱ソフトウェア

- アセンブラ
- リンカ
- ソースコードデバッガ
- システムモニタ
- フロートエミュレータ
- コマンドシェル

価格

- ボードパッケージ（XVI対応）
  VDTK-X68K…………¥248,000
- オプションソフト（Cコンパイラ）
  VDTK-C-X68K………¥68,000

オプションソフトウェア

- Cコンパイラ
  (VDTK-C-X68K)

購入方法

上記商品は当面の間、通信販売のみとさせて頂きます。購入ご希望の方は、住所、(社名、所属)氏名、電話番号をお知らせ下さい。注文書をお送りいたします。

※製作：ボード………有限会社アクセス
ソフトウェア……株式会社ハドソン

有限会社アクセス
〒101 東京都千代田区神田神保町1-64
神保町協和ビル7F
☎03(3233)0200(代) FAX.03(3291)7019

資料請求券 oh!X 2月号

パソコン/ワープロ通信ネットワークサービス

J&P HOT LINE

第1回　68さん　ID:JH122004

HOTLINEのX・MZシリーズユーザーにその魅力を語ってもらおうというネットワーカー・ネットワーク。今回は、J&PのCUGで活躍されている68さんの登場です。JTCは、68さんが加入されているCUG。ある会社が主催している、そこの社員でないと入れない秘密の花園。とびかうX68000の話題がずいぶんと楽しいようです。

基本データ

- ■使用機種名：CZ-611CBK
- ■所有周辺機器：コプロセッサー、イメージユニット 増設でメイン6M
- ■使用開始時期：1988年7月24日から
- ■おすすめX68000用フリーソフト：LHA.X、必需品ですね (SHARP-HOTLINE内にあり)
- ■X68000への希望：MS-DOSの、HC形式の正式サポート

■X68000購入の理由は？

当時フリーエリアが最も多い日本語パソコンだったから。
そのうえ、DOS／FEP／DOS-BASIC／スクリーンエディター標準添付。しかも、MS-DOSテキストファイルが読めたから。(OS-9も使えたし。)

■お気に入りのゲームソフトは？

光栄の「大航海時代」をJTCメンバーと共に盛り上がってやりましたね。通信のBBSを読んでいてこのゲームを知り購入。その後、港や特産品、貿易の儲け方等を電子メールやOLTで情報交換して、2カ月ぐらいこのネタで盛り上がりましたよ。X68が"提督港を発見しやしたぜ!!"と喋るのが楽しかったな。他にも現在入手不可能ですが、「A列車で行こう2」「A列車で行こう3」それに、「シムシティ」も忘れてはいけないかな。

■ビジネスで活用するソフトは？

一番用途が多いのは標準添付のED.XとASKです。「CARD-PRO」は、帳票画面を幾つか設定出来たし、出力においても出来たので、以前ログ管理や、家計簿、名刺管理、電話帳、仕事がらみで使用していましたが、今は殆ど使っていません。あと通信ソフトは、「た～みのる2」。ひととおりの機能が完備されていますから。用途は完全オートパイロットによるJTC閲覧保存です。

■X68000のよいところ、楽しい部分は？

ローマ字入力時「ん」がXキー、1回のキーストロークで入力可能な事。Nの後に子音を入れるかN2回のキーストローク必要機種が多いけれどX68000は違う。それから不意のOLT等でもメインメモリーの多さで、2M以上のジャーナルでも安心して通信ログを保存出来る事。また、ビープ音等にユーザーで好きな音、音声等を設定出来る事。起動時音声メッセージを出させたり楽しいですよ。

T1002179020604 雑誌 02179-2

Printed in Japan